ARMS Administrator Manual

Chris Dallimore.

Centre for Water Research
University of Western Australia.

Contents

Introduction

1.1 Aquatic Realtime Management System

ARMS (Aquatic Realtime Management System) is a decision support system to aid managers and operators of surface water bodies (reservoirs, lakes, rivers, estuaries, coastal oceans). Its underlying philosophy is to provide an automated software system that requires minimal maintenance to monitor and forecast the conditions of surface water resources, and to notify relevant staff of current conditions on a regular basis.

ARMS is an automated software package that manages historical and real-time water resource data, has a user friendly visualisation interface, posts information on the Internet, provides real-time and forecast numerical modelling capabilities, sends messages via email on the status of water resources, and computes decision support indices to aid in operational management. ARMS provides decision support over a variety of surface water resources (lakes, reservoirs, rivers, estuaries, marine coasts) to aid in the management of water supply, water quality, pollutant spills, flooding, and environmental lows. ARMS is an excellent decision support tool for water authorities, hydropower operators, and environmental regulators. ARMS functions on a variety of computing platforms and operating system environments.

1.2 Capabilities

ARMS was designed to require minimal human interaction and to provide useful information to water resource managers in a real-time manner. ARMS provides the following functionality:

1. Real-time data management from automated monitoring
2. Management of historical water resource databases
3. Real-time simulations of current conditions
4. Up-to-date forecast simulations
5. Email alerts of monitoring failures, incidents, and events
6. Web posting of summary status reports, simulations, and alerts
7. Quality control for data consistency
8. User-friendly visualisation of data and simulations

1.3 Benefits

ARMS provides the following benefits for water resources management:

1. Automated real-time monitoring and numerical forecasting of key operating parameters with an email service to alert staff of monitoring failures and incidents
2. Automated web postings of the current and forecast conditions of surface waters inclusive of decision support indices for sustainability
3. Assessment and control of intervention strategies for reservoir water quality such as destratification devices, withdrawal strategies, and chemical dosing
4. Scenario analysis to reduce nutrient, suspended sediment, and pathogens and optimise placement of water supply intakes in reservoirs, lakes, and rivers
5. Carbon sequestration estimates in lakes and reservoirs
6. Real-time monitoring and forecasting of river water quality below dams, tributaries, sewage treatment plants, and other point and distributed sources
7. TDML (Total Daily Maximum Load) monitoring and modelling of catchments and rivers
8. Real-time monitoring and forecasting, and rapid assessment of pollutant or toxic spills in surface waters
9. Assessment of reservoir withdrawal strategies to meet downstream fish habitat and environmental guidelines
10. Investigate options to reduce costs of water treatment
11. Increased government and public access to real-time water resources information
12. Water balance accounting for water supply purposes
13. Assessment of risk management strategies

1.4 Implementation

ARMS is written primarily in Java to allow for cross-platform installations. ARMS is compatible with a number of SQL (Structured Query Language). databases. Currently ARMS Version 2 is compatible with ORACLE, PostgreSQL, and Microsoft SQL Server (TM).

Conceptually ARMS is essentially a set of 'resources' which point to particular components of the ARMS database, to external SQL databases or to components of the file system (model output and input, image files etc). ARMS Data Resources may be 1 Dimension scalar time series, 2D Vector timeseries, Profile Timeseries data, individual profiles, or 1D, 2D or 3D temporally varying model output.

Processing of the 'resources' is carried out by 'tasks'. A wide variety of tasks are available for processing (quality checking, downsampling, averaging etc) and viewing the data.

Configuration of 'resources' and 'tasks' is done via a number of XML (Extensible Markup Language) files. XML is a generic framework for storing any amount of text or any data whose structure can be represented as a tree and allows the user to modify any properties that have been defined for a particular task or resource.

Initial setup of an ARMS installation is usually undertaken by CWR of HydroNumerics but this document aims to aid the user to carry out basic manipulation of the tasks and resources. The format of this guide will be to document the XML format required for the key ARMS resources and tasks. For all XML files the order of the elements is crucial and if any errors are found in the XML when loading, ARMS will throw an error giving the name and line number of the offending file. The help menu in the ARMS GUI displays information on a number of key tasks and gives the XML schema associated with the task. The schema in the help menu is automatically generated from the code and is therefore always in sync with the application and should be used if there are differences between this manual and the application.

The ARMS GUI has an inbuilt XML text file editor for viewing ARMS configuration files. Right clicking on any XML file in the text editor allows users to see the XML Schema and to check if the current text validates using the Schema.

1.5 Certification

ARMS requires the Java 1.6 run-time environment to run. It is recommended that the latest Java security updates are installed.

1.5.1 Operating Systems

ARMS is certified to run on the following operating systems:

- Max OSX 10.4+
- Windows XP Service Pack 2
- Windows 7
- Windows Server 2008 Service Pack 2
- Red Hat Linux 4.1.2

Provided the necessary Java run-time is installed most other operating system should also work.

1.5.2 Databases

ARMS is certified to run on the following SQL Compliant databases:

- PostgreSQL 8.3
- PostgreSQL 8.4
- Microsoft SQL Server 2000
- Microsoft SQL Server 2005

- Oracle 10g

Installation

2.1 Introduction

Once the ARMS Database has been generated (see Chapter 3) the ARMS application must be configured to connect to the required ARMS database and any addition SQL databases that it needs to access. The ARMS application is one executable file that operates as both the processing server and the client application. It is recommended that all client application instances are configured to use read only database connections.

2.2 ARMSWorkspace

ARMS makes use of a work space convention and assumes that the key database connection files are stored in the root directory of the current work space. The work space to be used is stored in the **WorkSpace** value in the **ARMSPreferences.xml** located in *$HOME/.arms* where *$HOME* is the users home directory. ARMS will automatically create a preference file when it is first run and ask the user for the location of the workspace directory. The workspace directory is referred to below as $ARMSWorkSpace$.

2.3 Database Connection

This section outlines the XML file format of the two files required for connection to one or more databases. The two files contain the JDBC Connection information and username password information and must be called $ARMSDBConnections.xml$ and $ARMSPasswordsRepository.xml$ and must be located in the current ARMS Workspace. The files should be created by the ARMS Administrator and distributed to the appropriate users.

2.3.1 ARMSDBConnections.xml

The $ARMSDBConnections$ File specifies the details for connecting to the database. The XML file has the following schema:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JDBCList>
  <comment></comment>
  <JDBC><!-- List -->
    <comment></comment>
    <name></name>
    <driver></driver>
    <url></url>
    <database></database>
    <armsDataBase>true</armsDataBase>
    <loadOnStartUp>false</loadOnStartUp>
    <passwordLookup></passwordLookup>
    <armsGUIFile>arms_gui.xml</armsGUIFile>
```

```
    <taskSchedulerListFile></taskSchedulerListFile>
    <startSchedulerOnStartUp>false</startSchedulerOnStartUp>
    <maxActiveConnections>10</maxActiveConnections>
    <maxWaitForConnection>5000</maxWaitForConnection>
  </JDBC>
  <HideMenus>
    <hideDBMenu>false</hideDBMenu>
    <hidePlotMenu>false</hidePlotMenu>
    <hideTaskMenu>false</hideTaskMenu>
    <hideWizardsMenu>false</hideWizardsMenu>
    <hideDYRESMMenu>false</hideDYRESMMenu>
    <hideELCOMMenu>false</hideELCOMMenu>
    <hideHelpMenu>false</hideHelpMenu>
  </HideMenus>
</JDBCList>
```

where

JDBC specifies the connection information to one database. More than one JDBC element can be specified.

name A name for this connection.

driver Specifies the type of database we are using. One of *org.postgresql.Driver* (PostgreSQL), *oracle.jdbc.driver.OracleD* (Oracle), *net.sourceforge.jtds.jdbc.Driver* (SQLServer).

url A valid JDBC URL pointing to the location of the database. Eg. jdbc:postgresql://server.cwr.uwa.edu.au/armsTesting

database The database name. Eg. armsTesting

armsDataBase true/false is this connection is to a ARMS Database. ARMS is able to access other SQL database using a Remote DB Resource.

passwordLookup The Key in *ARMSPasswordsRepository.xml* that hold the username/password information.

armsGUIFile A reference to an ARMS GUI Configuration file (see Chapter 4). This will be overridden if the user has a reference in the *USER_PROFILES.xml* file (see Section 3.5.1)

taskSchedulerListFile A reference to a ARMS Scheduler file for running tasklists at regular intervals.

startSchedulerOnStartUp true/false to control if the Scheduler should be started immediately of if it is to be started form the GUI.

maxActiveConnections and **maxWaitForConnection** JDBC connection configuration parameters. See JDBC documentation for more information.

The **HideMenu** allows users to hide particular menus in the ARMS GUI.

2.3.2 ARMSPasswordsRepository.xml

The *ARMSPasswordsRepository.xml* File specifies a list of username/password pairings for use by ARMS. Passwords need to be set for all JDBC Connections and for any other connections such as connecting to a FTP Server.

The XML file has the following schema:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<UsernamePasswordList>
  <PasswordRecord>
    <passwordLookup></passwordLookup>
    <UsernamePasswordPair>
      <username></username>
      <password></password>
    </UsernamePasswordPair>
  </PasswordRecord>
</UsernamePasswordList>
```

where

PasswordRecord specifies the key, username and password information for one connection. More than one PasswordRecord element can be specified.

passwordLookup is the key word for this record. This key should be used in all other configuration files when referencing a record.

username is the username to be used for the connection.

password is an encrypted password string. If the password element is not present then the user will be prompted to enter the password when a connection is first attempted at this stage the encrypted string can optionally be saved to the file.

2.4 ARMS Configuration Files

To simplify the configuration process for multiple ARMS users in an organisation ARMS Version 2 introduced the ability to store the XML configuration files in the database itself. This means that changes in the configuration can be rolled out to all users by simply updating the stored files in the database. The ARMS GUI provides the ability to interactively modify the stored database files. As an aside it should be noted that if desired, users can have locally modified version of one or more configuration file which will override the stored version. This allows users to modify attributes such as default plot axes or colour limits for their local setup.

The ARMS GUI shows all local and database stored configuration files in the Config Tree. The icon for each

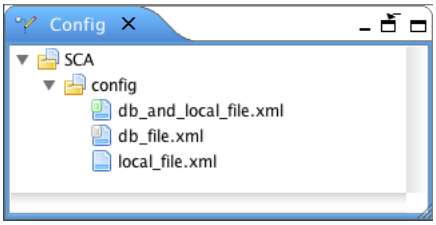

Figure 2.1 ARMS Config tree with icons showing locations of files

element shows where the configuration file if stored. A text file icon with a green database icon in the top left corner indicates the file is stored both locally and on the database (there may be differences between the two files). A text file icon with a grey database icon in the top left corner indicates the file is only stored in the database. A text file icon with no database icon indicates the file is only stored locally. Right-clicking on a configuration file allows a number of interactions to be performed on the file. These interactions include:

- Exporting and importing the file to and from the database
- Deleting either the local of database copies
- Performing a difference between local and database copies
- Validating the XML

ARMS Database

3.1 Introduction

ARMS is compatible with a number of SQL (Structured Query Language). databases. Currently ARMS Version 2 is compatible with ORACLE, postgreSQL and Microsoft SQL Server (TM). To optimise performance the ARMS database schema has been designed to make use of a large number of tables and stored SQL procedures. This chapter discusses the database schema and the key stored procedures.

Setup of an ARMS Database consists of two steps 1) Creation of the standard tables, stored procedures and default user roles 2) Importing of resources. The importing of resources creates the individual sensor repository tables for each of the ARMS sensor objects. If additional ARMS database resources are added to the ARMS configuration the import resource task needs to be run again.

3.2 Database Setup

Setup of an ARMS Database consists of two steps 1) Creation of the standard tables, stored procedures and default user roles 2) Importing of resources. The importing of resources creates the individual sensor repository tables for each of the ARMS sensor objects. If additional resources are added to the ARMS configuration the import resource task needs to be run again.

3.2.1 Database Initialisation

The database initialisation is carried out using the arms.jar file. This is done using the following command from a terminal.

```
java -cp arms.jar cwr.arms.tasks.database.init.InitialiseDBTaskFactory init.xml
```

Where **init.xml** is a XML configuration file with the following schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<InitialiseDB>
  <name></name>
  <driver></driver>
  <urlRoot></urlRoot>
  <url></url>
  <rootLookupKey></rootLookupKey>
  <writeLookupKey></writeLookupKey>
  <readLookupKey></readLookupKey>
  <initialiseDB>true</initialiseDB>
  <removeDB>false</removeDB>
</InitialiseDB>
```

where

name: gives the name of the database. This should be the same as the name in the arms.xml configuration file.

driver: is database driver. One of *org.postgresql.Driver* (PostgreSQL), *oracle.jdbc.driver.OracleDriver* (Oracle) or *net.sourceforge.jtds.jdbc.Driver* (SQLServer).

urlRoot: is the JDBC URL to the root database on the database server of the form $jdbc : postgresql : //server.url.ip/ < /urlRoot >$. This is not used for $ORACLE$.

url: is the JDBC URL to the new ARMS database on the database server of the form $jdbc : postgresql : //server.url.ip/name < /urlRoot >$. *name* should be consistent with the name given above.

rootLookupKey: is a reference key to a username/password lookup in the $ARMSPasswordsRepository.xml$ file located in the $ARMSWorkSpace$. If this is set to *prompt* the user will be asked for a username/password for a database super user.

writeLookupKey: is a reference key to a write username/password lookup in the $ARMSPasswordsRepository.xml$ file located in the $ARMSWorkSpace$. To ensure the write user has full access to all GUI database interaction the username should be the same as the database name.

readLookupKey: is a reference key to a read only username/password lookup in the $ARMSPasswordsRepository.xml$ file located in the $ARMSWorkSpace$.

initialiseDB: true/false indicating whether database should be initialised.

removeDB: true/false indicating whether database should be removed before initialisation.

NOTE: For Oracle initialisation the Oracle Database Administrator should first create an empty database with the correct database name.

3.2.2 Importing Resources

The importing of resources into the database is a standard ARMS task and can be run like any other task (see Chapter 6).

The configuration file for the task has the following schema

```
<?xml version="1.0" encoding="UTF-8"?>
<ImportDB>
  <rootAddress></rootAddress>
  <readLookupKey></readLookupKey>
  <writeLookupKey></writeLookupKey>
</ImportDB>
```

Where: **rootAddress**: is the root ARMS address below which all resources will be imported. Initially this should be the top level context but may be changed later to only import the sub-domain where changes have occurred. The only motivation for importing a sub-domain is to speed up the import process.

readLookupKey: is a reference key to a read username/password lookup in the *ARMSPasswordsRepository.xml* file located in the *ARMSWorkSpace*.

writeLookupKey: is a reference key to a write username/password lookup in the *ARMSPasswordsRepository.xml* file located in the *ARMSWorkSpace*.

3.3 Database Tables

3.3.1 Time Series Data

The major data storage component in ARMS is time series data. Time series data represents measured data at a particular location at a series of times. Time series data will often have a constant interval between measurements but there is no ARMS requirement for this and irregularly spaced data can be stored in the same table structure. Time Series data in ARMS can be 1-Dimensional (eg: shortwave radiation measured on a lake) or 2-Dimensional (a wind vector). A particular time series resource is referred to as a "sensor". A sensor may refer to either the raw data or to processed (e.g downsampled or averaged) data.

For performance reasons each time series sensor is stored in its own table. The name of the individual sensor data tables are of the form **sensor_repository_XXXXXX** where **XXXXXX** is a 6 digit integer sensor identification code (**SID**). The list of sensors and their **SID** codes is kept in the **sensors** table. The columns in the **sensors** table are shown in Table 3.1. The columns in the **sensor_repository_XXXXXX** table are shown in Table 3.2.

The final database table of importance for timeseries data is the **sensor_jdn_boundaries** table. This table stores the date limits of all sensor resources and is updated when ever data is written to a sensor repository table. The motivation for this table is to reduce the load on the database as there is a variety of times when the date limits are required.

Table 3.1 Summary of columns in the sensors table

Column Name	Description
s_code	The Sensor ID code
s_name	The ARMS Address for this sensor of the form site.station.data.sensor. The address of a particular resource matches up to its location in the GUI tree.
s_description	A description of the sensor. Currently this defaults to the ARMS address but may be used in future enhancements.
s_resource_code	A link to the resources table that associates this sensor with a particular ARMS context.
s_type_code	A reference to group particular sensors together by type. Currently each sensor has a unique type code
s_table_name	The table containing the data for this sensor
s_table_type	The type of data being store. Currently one of RAW, SAMPLED or AVERAGED
s_x_offset	Offset from the datum in the x direction m
s_y_offset	Offset from the datum in the y direction m
s_z_offset	Offset from the datum in the z direction m
s_datum_code	The datum code (in the **datums** table that this sensors position is relative to
s_datum_value	Currently unused
s_graph_value	A short string that can be used when plotting data
s_min_value_1	The minimum expected value of this sensor used in Quality Checking task
s_max_value_1	The minimum expected value of this sensor used in Quality Checking task
s_min_value_2	Currently unused
s_max_value_2	Currently unused
s_min_value_3	Currently unused
s_max_value_3	Currently unused
s_min_value_4	Currently unused
s_max_value_4	Currently unused
s_min_value_5	Currently unused
s_max_value_5	Currently unused

Table 3.2 Summary of columns in the sensors repository tables

Column Name	Description
st_code	unique index for this measurement
st_sensor_code	the sensor ID code
st_sensor_upload_code	Currently unused
st_feed_date_jdn	the Julian date of this measurement. The Julian date (JD) is the interval of time in days and fractions of a day since January 1, 4713 BC Greenwich noon in UTC. All dates are therefore stored in a UTC time zone
st_x_offset	the x offset from the station location
st_y_offset	the y offset from the station location
st_z_offset	the height of this measurement above the main datum (nominally AHD for Australian sites)
st_datum_code	The datum code (in the **datums** table that this sensors position is relative to
st_value_1 double	The value of the measurement for this date. If the sensor is 2D this is the value in the X direction
st_value_2 double	If the sensor is 2D this is the value in the y direction
st_quality_code_1	The quality code (see Table 3.3) of the measurement for this date.If the sensor is 2D this is for the value in the x direction
st_quality_code_2	If the sensor is 2D this is for the value in the y direction

Table 3.3 List of quality Code values

Quality Code	Description
0	NULL - No quality code has been set
1	RAW - Imported Raw data that has not been quality checked
2	QC_VALID - Data passed quality checking task
3	DOWNSAMPLED - Data has been downsampled from data that passed quality checking
4	FILLED - Data has been filled from nearby data using the time series filling task
5	FILLED_DEFAULT - Data has been filled with a default value
6	AVERAGED - Data has been averaged
7	DERIVED - Data is derived from other data streams
8	QC_INVALID - Data failed quality checking task
9	Profile Data
10	ELCOM Data
11	DYRESM Data

Table 3.4 Summary of columns in the sensors time boundaries table

Column Name	Description
sjb_code serial	unique index for this entry
sjb_sensor_code	the sensor ID code
sjb_min_jdn	the Julian date of the first measurement
sjb_max_jdn	the Julian date of the last measurement

3.3.2 Profiler Data

The second form of data that ARMS stores is data collected from profiling instruments. Every time a profiler file is imported into the database a new **profile_repository_XXXXXXX** table is where **XXXXXXX** is a 7 digit identification code. Each profile repository table contains all the measurements collected in one profile and any processed data for the profile. The list of profiles and their **ID** codes is kept in the **profiles** table. The columns in the **profiles** table are shown in Table 3.5. The columns in the **profile_repository_XXXXXXX** table are shown in Table 3.6.

3.3.3 Imported Files

ARMS is able to import a range of raw data files into the ARMS database. To ensure that data is not imported twice two tables have been setup to keep track of imported files. The **files** database table links the file locations to a file code identification number. The **file_list** table is the complete list of all files that have been imported. If for any reason a file needs to be re-imported the user simply needs to delete the relevant entry from the **file_list** table and rerun the file processing task list. The columns of these two tables are shown in The columns in the 3.7 and 3.8.

Table 3.5 Summary of columns in the profiles table

Column Name	Description
p_code	The profile ID code
p_name	The name of this profile currently set to the root address for the profiler resource
p_description	A description of the profile
p_resource_code	A link to the resources table that associates this sensor with a particular ARMS context.
p_table_name	The table containing the data for this profile
p_latitude	The latitude at which the profile was collected
p_longitude	The longitude at which the profile was collected
p_utm_x	The UTM Easting at which the profile was collected
p_utm_y	The UTM Northing at which the profile was collected
p_utm_zone	The UTM Zone in which the profile was collected
p_date_jdn	The Julian date at which the profile was collected
p_datum_code	The datum code (in the **datums** table that this sensors position is relative to
p_datum_value	The offset from the given datum

Table 3.6 Summary of columns in the profile repository tables

Column Name	Description
pt_code	A unique ID for this measurement
pt_profile_code	The profile ID code
pt_address	The ARMS address for the profile that this measurement is part of.
pt_x_offset	The offset in the x direction for this measurement
pt_y_offset	The offset in the y direction for this measurement
pt_z_offset	The depth of the measurement
pt_value	The value of the measurement
pt_quality_code	The quality code (see Table 3.3) of the measurement

Table 3.7 Summary of columns in the files table

Column Name	Description
f_code	A unique code for files imported from this path
f_path	The path to the imported files

Table 3.8 Summary of columns in the file_list tables

Column Name	Description
fl_code	A unique code for this file
fl_file_code	A link to the file_list table
fl_value	The name of the imported file

3.3.4 Configuration Files

To simplify the configuration process for multiple ARMS users in an organisation ARMS Version 2 introduced the ability to store the XML configuration files in the database itself. This means that changes in the configuration can be rolled out to all users by simply updating the stored files in the database. The ARMS GUI provides the ability to interactively modify the stored database files. As an aside it should be noted that if desired, users can have locally modified version of one or more configuration file which will override the stored version. This allows users to modify attributes such as default plot axes or colour limits for their local setup. The columns in the **config** table are shown in Table 3.9.

Table 3.9 Summary of columns in the config table

Column Name	Description
c_filename	the path of the file relative to the ARMS Workspace e.g: /site/config/arms.xml
c_content	the contents of the file
c_xmltype	the XMLRoot element type this is automatically generated from the file content when files are exported to the database

3.3.5 Additional Tables

A short description of the tables in the database not covered above is given in Table 3.10.

Table 3.10 Summary of database tables

Column Name	Description
application_errors	A table to hold a list of application errors. Currently unused
datums	A list of the various height datums used in the configuration.
diagnostics_log	A log of warnings and errors encountered during quality checking of data
error_codes	A list of error codes and messages used by ARMS
locations	Stores geographical information about the various station sites in the configuration. Legacy table now obsolete
models	A table to store model configuration text file. Currently unused
profile_task_status	A list of what processing has occurred on profiler data
resource_groups	A list of the different types of data groupings (e.g. raw or downsampled 15 min) that are used
resources	Stores geographical information about the various station sites in the configuration
savestamp	A timestamp of the latest write to the database
security_permissions	A table controlling which users have access to which table
security_users	List of the various user roles
sensor_types	A list of all the different sensor types
sensor_uploads	Currently unused
sites	Currently unused
time_zones	A map of basic time zones

3.4 Database Stored Procedure

This section is designed to give Database Administrators a brief overview of the stored procedures in an ARMS database. There should be little need to call these stored procedures from outside ARMS and care should be taken to ensure the user is aware on the consequences of running any function. Tables 3.11 - 3.16 gives summaries of the various categories of stored procedures.

Table 3.11 Summary of stored procedures used in initialisation

Procedure	Description
sp_create_datum	Creates a datum entry in the datums table
setDatum	Either create a datum using sp_create_datum or update an existing datum
sp_create_resource	Create a resource entry in the resource tree
setResource	Either create a resource using sp_create_resource or update an existing resource
sp_create_sensor	Create a sensor entry in the sensors table and create the sensor repository table
setResource	Either create a sensor using sp_create_sensor or update an existing sensor
sp_create_timezone	Create a time zone entry in the time zones table
setTimezone	Either create a time zone using sp_create_timezone or update an existing sensor
sp_util_init	Carries out initialisation of default resource groups and timezones
sec_init_user	Initialise user/role and set default permissions

Table 3.12 Summary of stored procedures used for reading/writing timeseries data

Procedure	Description
getProcessStartDate	Returns the suggested start date for a given resource. This returns the date of the 1st NaN, or the last date + timestep
getAllSid	get all sensor ids and their associated ARMS address
getTimeSeriesStart	get the first date of the given sensor
getTimeSeriesEnd	get the end date of the given sensor
getSensorSeries	get all the required information to create an ARMS timeseries from a particular sensor between the given dates
getQualityCount1	returns the number of data items for the given sensor between the given dates with the given quality code
setSensorSeries	add or update the data for a given sensor
setTimeSeriesBoundaries	refresh the **sensor_jdn_boundaries** table for the given sensor by querying the database for the start and end dates.
setInvalidQualityCodes	set the quality code to QC_INVLAID for the given sensor between the given dates
setReplaceQualityCodes	Performs a bulk change of QualityCodes from x to y
setConstantHeight	Set all heights/z_offset between two date for the given sensor to the given value
setDynamicHeight	Set all heights/z_offset between two date for the given sensor using the z_offset and another resource this routine is no longer used due to performance issues
getNumberOFElementsBetween	Count the number of records point between two date
deleteData	delete all data for the given sensor between the given dates

Table 3.13 Summary of stored procedures used for reading/writing profiler data

Procedure	Description
setProfileTable	Dynamically creates a profile repository table
setProfile	Inserts or updates a reference in the profiles table
setProfileSeries	Creates the profile data in a particular profile repository table
setProfilerQC	Set the quality codes for profiler data
getProfile	Obtain the profile data of a particular profile ID
getProfileByDate	Get profile data at a given date
getProfilesBetweenDate	Get all profile data between two given date
getProfiles	Returns a complete profile hash map of all profiles
getFirstProfileDate	Returns first profile in a given address
getLastProfileDate	Returns last profile in a given address
getFirstProfileDateByUtm	Returns first profile in a given address and within a given UTM Rectangle
getLastProfileDateByUtm	Returns last profile in a given address and within a given UTM Rectangle
getProfilesByAddress	Returns a complete profile hash map of all profiles in a given address
getProfilesByArea	Returns all profiles within a given UTM Rectangle
getProfilesByAreaAndTime	Returns all profiles within a given UTM Rectangle and between start and end date
getProfileSeries	Return a profile series between 2 depths

Table 3.14 Summary of stored procedures used for reading/writing configuration files

Procedure	Description
getConfigs	returns all configuration file stored in the database
setConfigFile	update/add a new configuration file
getConfig	get the configuration file with the given file path
getConfigsLike	get all configuration files with a file path like the given path

Table 3.15 Summary of stored procedures used for accessing imported files information

Procedure	Description
getList	returns the list of imported files from the given file path
clearList	deletes all records from the file_list table with the given file path
setList	add a new record to the file_list table

3.5 Database User Roles

By default the database initialisation process creates two database roles for accessing the database, a root user and a read only user. Additional roles can be created using the standard procedures for creating roles in the particular database environment being used. Once the user has been created the security permission need to be set using the *sec_init_user* stored procedure as follows for a write user

```
select * from sec_init_user('new_write_user', true, true, true, true);
```

or for a read user

```
select * from sec_init_user('new_read_user', false, false, false, false);
```

3.5.1 Modifying GUI For Different Users

The database administrator can create different GUI configurations for different user roles by using the $USER\_PROFILES.xml$ file which must be stored in the database in the $config$ table with a path of $/dbname/config/USER\_PROFILES.xml$. This file is the only configuration file that will not be overridden by a local copy. This ensures the administrator can control the access of resources for all users.

This file allows the administrator to set different ARMS Root Config and ARMS GUI Config files and to hide various GUI components for different users. The schema for the file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<UserProfiles>
  <UserProfile><!-- 1 or more elements -->
    <username></username>
    <armsConfigFile>arms.xml</armsConfigFile>  <!--Optional-->
```

Table 3.16 Summary of miscellaneous stored procedures

Procedure	Description
setSaveStamp	set the date of the last write activity to the ARMS database
getSaveStamp	get the date of the last write activity to the ARMS database

```
    <armsGUIFile>arms_gui.xml</armsGUIFile>  <!--Optional -->
    <hideAllConfigs>false</hideAllConfigs>  <!--true/false Optional -->
    <hideConfigsLike></hideConfigsLike> <!--Optional List of String-->
  </UserProfile>
</UserProfiles>
```

where

username: is the username of the role to configure.

armsConfigFile: is the path to the ARMS configuration file to use for this user. This defaults to *arms.xml*. By having different ARMS configuration files the administrator can control which resources a user has access to.

armsGUIFile: is the path to the ARMS GUI configuration file to use for this user. This defaults to *arms_gui.xml*. The GUI configuration file allows users to have different plot and/or task menu items in their GUI. For information on the ARMS GUI file see Chapter 4.

hideAllConfigs: is this is set to true then then no Configuration files will be shown in the GUI Tree.

hideConfigsLike: One or more **hideConfigsLike** can be specified to hide a subset of the configuration files in the GUI tree. For example $< hideConfigsLike > task < /hideConfigsLike >$ will hide all files that have task in the file path.

GUI Configuration

4.1 Introduction

This section outlines the XML file format for controlling the configuration of the ARMS GUI. This file setups the look of the GUI, specifies the main ARMS control files and initialises any custom menus for running a list of tasks. The xml schema for the file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ARMSGui>
  <comment></comment>
  <armsConfigFile>arms.xml</armsConfigFile>
  <title></title> <!--Obsolete-->
  <aboutImage></aboutImage>  <!--Obsolete-->
  <summaryFile></summaryFile>
  <taskRootDir></taskRootDir>
  <TaskMenu>
    <address></address>
    <taskListFile></taskListFile>
  </TaskMenu>
  <plotRootDir></plotRootDir>
  <PlotMenu>
    <address></address>
    <plotFile></plotFile>
  </PlotMenu>
</ARMSGui>
```

where

comment is a optional String containing any comment.

armsConfigFile The name of the main ARMS configuration file. This file **must** be in the same directory as the GUI configuration file.

title Obsolete.

aboutImage Obsolete.

taskRootDir A relative path to the root directory for task list files. If not specified task list files are relative to the location of GUI configuration file.

taskMenu One or more sub-elements for setting up the Task menu item of the GUI. Each sub element has two elements **address** and **taskListFile**. The format of the address string is a series of names separated by dots (.). Each name represents a tree item for the menu (under the main Tasks menu item). So an address of 'LDS.T1.Run Processing' will generate a menu item as shown in Figure 4.1. The location of the task list file is relative to *taskRootDir* given above. For more information of the format of the task list file see Section 6.2.

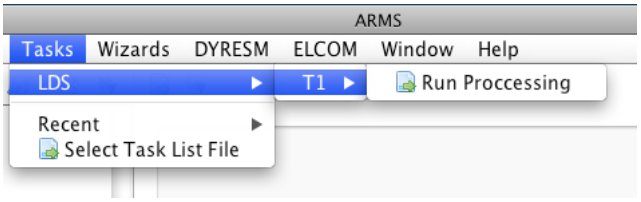

Figure 4.1 Example Task Menu Item for address LDS.T1.Run Processing

plotMenu One or more sub-elements for setting up the Plots menu item of the GUI. Each sub element has two elements **address** and **plotFile**. The format of the address string is the same as for taskMenu elements above. The location of the *plotFile* is relative to *plotRootDir* given above. Plot Files can be *CustomPlot*, *ProfilePlot* or *MoviePlot* files.

CHAPTER 5

Resources

5.1 Introduction

Conceptually ARMS can be thought of as a portal to a number of resources that give access to various data streams. The tree has a hierarchical structure with different nodes in the hierarchy representing different data streams. the ARMS configuration links each node to a specific concrete database technology, for example, a networked SQL database, a model output file or local text file. This approach allows all the data involved in the system to be navigated and investigated in a single simple user interface despite the heterogeneous and complex nature of the underlying database systems. This section details the XML configuration files that are used to create the ARMS Resource tree.

Currently there are two major ARMS resource generation factories. 1. Station 2. ELCOM

A station resource is generally considered to be a set or one or more sensors that are grouped together due to their physical location. For example a Lake Diagnostic System or a river gauging station. The station configuration initialises resources for raw and processed data for each of the individual sensors and can group sensors together to form a Profile Time Series resource which allows access to all sensors in a chain of sensors such as a thermistor chain.

A profiler resource is a special instance of a station resource and provides access to all data collected with a profiling instrument on one site such as a Lake. Methods are available for plotting individual profiles and for looking at where profiles have been collected.

A ELCOM resource is a wrapper around all the input and output files for a ELCOM-CAEDYM model simulation. A simulation may be a hindcast, forecast or particular scenario.

5.2 Station Resources

A station resource configuration needs to provide information about the name of the station, the location of the station and the sensors on the station. It also provides information on the resource groups that the data will be processed into. Due to the large size of the XML schema for a station resource the schema will be broken into sections for discussion

5.2.1 Header

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Station>
  <comment></comment>
  <name></name>
```

```
<title></title>
<longname></longname>
<site></site>
<latitude>0.0</latitude>
<longitude>0.0</longitude>
<timeZone></timeZone>
<daylightSavingsAware>false</daylightSavingsAware>
<surfaceDatum></surfaceDatum>
<bottomDatum></bottomDatum>
<datum></datum>
<zOffset></zOffset>
```

The header section contains the following elements

name the name of the station

title a title for display purposes

longname a title for display purposes

site the name of the site this station is located in

latitude the latitude of the station in decimal degrees

longitude the longitude of the station in decimal degrees

timeZone the timezone for this station. This can be an offset from GMT in hours such as *-2.5* or a full name such as *America/Los_Angeles*. If no timeZone is given the timeZone form the main ARMS configuration file will be used.

daylightSavingsAware does data collected at this stage know about daylight savings or is it always collected at local standard time.

surfaceDatum a default surface datum for profile groups. Can be overridden for individual profiles

bottomDatum a default surface datum for profile groups. Can be overridden for individual profiles

datum a default datum for sensors. Can be overridden for individual sensors

zOffset a default datum offset for sensors. Can be overridden for individual sensors

5.2.2 Resource groups

```
<StationResourceGroup>
  <address></address>
  <longname></longname>
  <processingInterval>0</processingInterval>
  <processingOffset>0</processingOffset>
</StationResourceGroup>
```

StationResourceGroup used to define different aggregates of the station data. For examples that can be used to say that the station will have three types of data raw, downsampled and averaged. Each group will then have the same tree structure as set out in the sensor configuration below. The elements of **StationResourceGroup** are

address The arms address of the resource group relative to the station. For example *raw* or *downsampled*.15*min*

longname A display name for the group

processingInterval The expected data period in seconds

processingOffset An offset from midnight in seconds. For example if we have 30 minute data at 5 minutes and 35 minutes past the hour the **processingInterval** will be 1800 and the offset 300.

5.2.3 Sensors

The **StationSensors**, **StationSensorsGroup** and **StationProfileGroup** all have a similar layout as they all are wrappers for one or more sensors. The only difference between a **StationSensors** and a **StationSensorsGroup** is that a **StationSensorsGroup** is just a convenient way to group a number of sensors together in one folder in the tree structure. For instance this is often used to group all meteorolgical sensors together. A **StationProfileGroup** will also group all sensors in a folder but knows that all sensors form part of a chain of sensors that provide a profile of the water column. For instance a group of thermistor sensors at different depths that make up a thermistor chain. **NOTE:** All sensors in a profile group should be sampling at the same frequency.

The schema for the three sensor wrappers are:

```
<StationSensors>
  <longname></longname>
  <title></title>
  <sensorsFile></sensorsFile>
  <SensorInfo></SensorInfo>
  <PlotProperties></PlotProperties>
  <ignoreGroup></ignoreGroup>
</StationSensors>
<StationSensorsGroup>
  <name></name>
  <longname></longname>
  <title></title>
  <sensorsFile></sensorsFile>
  <SensorInfo></SensorInfo>
  <PlotProperties></PlotProperties>
  <ignoreGroup></ignoreGroup>
</StationSensorsGroup>
<StationProfileGroup>
  <name></name>
  <longname></longname>
  <title></title>
  <sensorsFile></sensorsFile>
```

```
    <SensorInfo></SensorInfo>
    <PlotProperties></PlotProperties>
    <ignoreGroup></ignoreGroup>
  </StationProfileGroup>
```

Where

name is the name of the sensor group or profile group

longname is a longer descriptive name for the group

title is the title for display purposes

sensorsFile is one or more links to files containing sensor information see Section 5.2.6

SensorInfo an individual sensor information element 5.2.6

PlotProperties the plot properties element controls a number of plotting functions. See Section 8.5 for a complete description.

ignoreGroup a list of resource groups that this sensor or group will not be a part used for. The link is to the **address** in one of the **StationResourceGroups** above. For example, this can be used to not have a derived sensor in the raw group.

5.2.4 Profiler

A profiler resource is a resource of data collected from a profiling instrument i.e. an instrument that collects data as it falls through the water column. The schema for a **StationProfilerGroup** is

```
<StationProfilerGroup>
  <comment></comment>
  <name></name>
  <longname></longname>
  <title></title>
  <sensorsFile></sensorsFile>
  <SensorInfo></SensorInfo>
  <PlotProperties></PlotProperties>
  <ignoreGroup></ignoreGroup>
  <bathymetryResource></bathymetryResource>
  <StationLocation>
    <comment></comment>
    <name></name>
    <latitude>0.0</latitude>
    <longitude>0.0</longitude>
    <searchRadius>0.0</searchRadius>
  </StationLocation>
  <Transect>
    <comment></comment>
    <name></name>
    <station></station>
    <searchTime>0.0</searchTime>
  </Transect>
```

```
</StationProfilerGroup>
```

The schema is similar to the **StationSensorsGroup** and **StationProfileGroup** but with the following additional elements.

bathymetryResource is an ARMS address to an ELCOM bathymetry resource. This allows the locations of profiles to be plotted on a map of the reservoir. The ELCOM bathymetry file needs to be correctly geo-referenced.

the **StationLocation** and **Transect** elements are for future enhancements to ARMS to allow profiles at a particular location or along a particular transact to be grouped together and plotted.

5.2.5 Withdrawal Gate

A withdrawal gate resource is a special resource for selective withdrawal systems. It essentially creates three 1-Dimension timeseries resources, One for the top height of the gate, one for the top height of the gate, and one for the status (open/closed).

The schema for the withdrawal gate resource is

```
<WithdrawalGate>
  <title></title>
  <minHeightTitle></minHeightTitle>
  <maxHeightTitle></maxHeightTitle>
  <statusTitle></statusTitle>
  <units></units>
  <datum></datum>
  <minHeightValue>0.0</minHeightValue>
  <maxHeightValue>0.0</maxHeightValue>
  <defaultHeightValue>0.0</defaultHeightValue>
  <defaultStatusValue>0</defaultStatusValue>
  <PlotProperties></PlotProperties>
  <ignoreGroup></ignoreGroup>
</WithdrawalGate>
```

title is the title for display purposes of the withdrawal gate resource;

minHeightTitle, **maxHeightTitle** and **statusTitle** are the titles for each of the sub resources

units is the height unit measurement

datum is the reference name of a datum in the Level Linker Resource (section 5.5)

minHeightValue and **maxHeightValue** are range values for quality checking of the height resources.

defaultHeightValue and **defaultStatusValue** are default values to be used by filling tasks.

PlotProperties and **ignoreGroup** are as above

5.2.6 Sensor Information and Sensor Files

A **SensorInfo** element in a station resource provides information on one sensor. Where a sensor is usually a particular measurement device. A **SensorFile** is simple a collection of different sensor elements which allows multiple stations to use the same information. For example all Lake Diagnostic System will usually have the same meteorological instruments so we can create one **sensorsFile** which is linked to by all of the LDS Station resources.

A **SensorFile** has the following schema

```
<SensorFile>
  <Sensor1D></Sensor1D>
  <Sensor2D></Sensor2D>
  <SensorScript></SensorScript>
  <SensorRemote1D></SensorRemote1D>
</SensorFile>
```

Where each sub-element can have 0 to n instances.

Sensor1D elements are 1-Dimensional ARMS database resource. The schema is

```
<Sensor1D>
  <name></name>
  <longname></longname>
  <SensorInfo></SensorInfo>
  <PlotProperties></PlotProperties>
</Sensor1D>
```

Sensor2D elements are 2-Dimensional ARMS database resource. The schema is

```
<Sensor2D>
  <name></name>
  <longname></longname>
  <SensorInfo></SensorInfo>
  <PlotProperties></PlotProperties>
</Sensor2D>
```

SensorScript elements are script resource. Script Resources uses javascript to create a resource derived on one or more other ARMS resource. The schema is

```
<SensorScript>
  <name></name>
  <longname></longname>
  <SensorInfo></SensorInfo>
  <ScriptBlock>
    <ScriptStream>
      <name></name>
      <address></address>
    </ScriptStream>
    <script></script>
  </ScriptBlock>
```

```
  <PlotProperties></PlotProperties>
</SensorScript>
```

The script block element contains two components. The **ScriptStream** is a keyword **name** - **address** pair where the **name** is a key to be used in the script and the **address** is an ARMS address pointing to a ARMS resource. To facilitate using the same **SensorScript** across different stations there are three keywords that can be placed in the address string. The key '$STATION', will automatically be replaced with the appropriate station name and the key '$RESOURCEGROUP' with the correct resource group. The key '$PARENT', will be replaced by the address of the parent of the script resource.

The **script** element contains the java script code. The easiest way to explain how the **SensorScript** operates is via an example

```
<ScriptBlock>
  <ScriptStream>
    <name>netrad</name>
    <address>lds.$STATION.$RESOURCEGROUP.met.netrad</address>
  </ScriptStream>
  <ScriptStream>
    <name>swrad</name>
    <address>lds.$STATION.$RESOURCEGROUP.met.swrad</address>
  </ScriptStream>
  <script>
    Math.min(Math.max(netrad - (0.92* swrad),-100),100);
  </script>
</ScriptBlock>
```

This script calculates the net radiation entering a reservoir using the measured Net and Shortwave radiation and then limits the value to between +/- 100;

SensorRemote1D elements are resource that link to a non-ARMS SQL database.

5.3 ELCOM Resources

A ELCOM resource configuration needs to provide information about the input and output files for a ELCOM-CAEDYM simulation. Due to the large size of the XML schema for a ELCOM resource the schema will be broken into sections for discussion

5.3.1 Header

The header gives some basic information on the particular ELCOM simulation. The schema is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ELCOMConfig>
  <directory></directory>
  <timeZone></timeZone>
  <datum></datum>
```

Where

directory is the base level for the simulation. This is usually an ELCOM run directory relative to the main ARMS work space for example *models/elcom/site/realtime*.

timeZone is the time zone the simulation is being run in. By default ARMS uses UTC time for all simulations. This ensures that all data and model files have a consistent time stamp. If desired this can be overridden for individual files using the sub-elements.

datum is an optional level linker datum. If not specified it is assumed the ELCOM bathymetry is relative to the ARMS main datum.

Below the header there are a number of sections with a similar format for linking to different types of files. These sections all have the following elements.

directory a directory name relative to the main directory in the header. Such as *infiles* or *outfiles nc*.

filename the name of the file(s) being referenced. For ELCOM output NetCDF files ARMS has the ability to link across multiple files generated from a series of restarts. This is achieved by using a wildcard "*" in the filename. For example *LDSProfile*.nc*

ELCOMVariable is a sub-element giving the details for one ELCOM input or output variable. The format of the ELCOMVariable schema is discussed below.

SensorScript is a sub-element giving the details for a java script derived variable. The format and methodology for script variables is the same as for the **SensorScript** elements in a **Station** resource.

elcomVariableFile is a reference to another configuration file containing a list of **ELCOMVariable** and **SensorScript** elements. Because the same output variables are often used for all ELCOM output files this is often the preferred method for defining the variables.

5.3.2 BC Files

Linking to ELCOM temporal boundary condition files is done using the **ELCOMBC** the sub-element schema is

```
<ELCOMBC>
  <directory></directory>
  <fileName></fileName>
  <name></name>
  <refNumber></refNumber>
  <timeZone></timeZone>
  <ELCOMVariable></ELCOMVariable>
  <SensorScript></SensorScript>
```

```
  <elcomVariableFile></elcomVariableFile>
</ELCOMBC>
```

refNumber refers to the boundary condition reference number.

5.3.3 Curtain File

Linking to ELCOM curtain NetCDF output files is done using the **ELCOMCurtain** the sub-element schema is

```
<ELCOMCurtain>
  <comment></comment>
  <directory></directory>
  <fileName></fileName>
  <name></name>
  <timeZone></timeZone>
  <ELCOMVariable></ELCOMVariable>
  <SensorScript></SensorScript>
  <elcomVariableFile></elcomVariableFile>
</ELCOMCurtain>
```

5.3.4 Sheet Files

Linking to ELCOM sheet NetCDF output files is done using the **ELCOMSheet** the sub-element schema is

```
  <ELCOMSheet>
  <comment></comment>
  <directory></directory>
  <fileName></fileName>
  <name></name>
  <timeZone></timeZone>
  <ELCOMVariable></ELCOMVariable>
  <SensorScript></SensorScript>
  <elcomVariableFile></elcomVariableFile>
</ELCOMSheet>
```

5.3.5 Aggregate Sheet Files

Linking to ELCOM aggregate sheet NetCDF output files is done using the **ELCOMAggregateOfSheet** the sub-element schema is

```
<ELCOMAggregateOfSheet>
  <comment></comment>
  <directory></directory>
  <fileName></fileName>
  <name></name>
  <timeZone></timeZone>
  <ELCOMVariable></ELCOMVariable>
  <SensorScript></SensorScript>
  <elcomVariableFile></elcomVariableFile>
</ELCOMAggregateOfSheet>
```

5.3.6 Profile Files

Linking to ELCOM profile NetCDF output files is done using the **ELCOMProfile** the sub-element schema is

```
<ELCOMProfile>
  <comment></comment>
  <directory></directory>
  <fileName></fileName>
  <name></name>
  <ELCOMProfileStation>
    <name></name>
    <i></i>
    <j></j>
  </ELCOMProfileStation>
  <timeZone></timeZone>
  <ELCOMVariable></ELCOMVariable>
  <SensorScript></SensorScript>
  <elcomVariableFile></elcomVariableFile>
</ELCOMProfile>
```

ELCOMProfileStation sub-elements give the **i** and **j** values of one profile location of one profile output in the file. The **name** is a reference used in the ARMS tree structure.

5.3.7 Aggregate Profile Files

Linking to ELCOM aggregate profile NetCDF output files is done using the **ELCOMAggregateOfProfile** the sub-element schema is

```
<ELCOMAggregateOfProfile>
  <comment></comment>
  <directory></directory>
  <fileName></fileName>
  <name></name>
  <ELCOMProfileStation>
    <name></name>
    <i></i>
    <j></j>
  </ELCOMProfileStation>
  <timeZone></timeZone>
  <ELCOMVariable></ELCOMVariable>
  <SensorScript></SensorScript>
  <elcomVariableFile></elcomVariableFile>
</ELCOMAggregateOfProfile>
```

ELCOMProfileStation sub-elements give the **i** and **j** values of one profile location of one profile output in the file. The **name** is a reference used in the ARMS tree structure.

5.3.8 Outflow Files

Linking to ELCOM outflow NetCDF output files is done using the **ELCOMOutflow** the sub-element schema is

```
<ELCOMOutflow>
  <comment></comment>
  <directory></directory>
  <fileName></fileName>
  <name></name>
  <bc>0</bc>
  <timeZone></timeZone>
  <ELCOMVariable></ELCOMVariable>
  <SensorScript></SensorScript>
  <elcomVariableFile></elcomVariableFile>
</ELCOMOutflow>
```

bc refers to the boundary condition reference number.

5.3.9 General 3D Files

Linking to ELCOM general 3D NetCDF output files is done using the **ELCOMGeneral3D** the sub-element schema is

```
<ELCOMGeneral3D>
  <comment></comment>
  <directory></directory>
  <fileName></fileName>
  <name></name>
  <i>0</i>
  <j>0</j>
  <k>0</k>
  <timeZone></timeZone>
  <ELCOMVariable></ELCOMVariable>
  <SensorScript></SensorScript>
  <elcomVariableFile></elcomVariableFile>
</ELCOMGeneral3D>
```

The **i**, **j** and **k** values give the location of one cell output in the file.

5.3.10 Drifter Files

Linking to ELCOM drifter NetCDF output files is done using the **ELCOMDrifter** the sub-element schema is

```
<ELCOMDrifter>
  <comment></comment>
  <directory></directory>
  <fileName></fileName>
  <name></name>
```

```
    <drifterNumber>0</drifterNumber>
    <timeZone></timeZone>
    <ELCOMVariable></ELCOMVariable>
    <SensorScript></SensorScript>
    <elcomVariableFile></elcomVariableFile>
  </ELCOMDrifter>
```

drifterNumber refers to the drifter number.

5.3.11 Common Configuration

Often in ARMS there will be more than one ELCOM-CAEDYM simulations for one site. For example a realtime simulation, a number of scenarios and a number of hind cast simulations for historical events. To ease the burden of generating multiple configuration files with essentially the same data the ELCOM configuration file can reference a file with a list of all the sub elements above using the **elcomCommonConfigFile** element.

For example we can have two ELCOM config files like

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ELCOMConfig>
  <comment>ELCOM Realtime</comment>
  <directory>models/elcom/lake/realtime/</directory>
  <timeZone>0</timeZone>
  <elcomCommonConfigFile>config/elcom/lake/common_files.xml</elcomCommonConfigFile>
</ELCOMConfig>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ELCOMConfig>
  <comment>ELCOM Realtime</comment>
  <directory>models/elcom/lake/flood_scenario/</directory>
  <timeZone>0</timeZone>
  <elcomCommonConfigFile>config/elcom/lake/common_files.xml</elcomCommonConfigFile>
</ELCOMConfig>
```

the config/elcom/lake/common_files.xml will then contain all the sub-elements such as **ELCOMBC** or **ELCOMSheet**.

5.3.12 ELCOMVariable

The schema for **ELCOMVariable** linked to in the various ELCOM file configurations is

```
<ELCOMVariable>
  <name></name>
  <refNumber></refNumber>
  <title></title>
  <units></units>
  <PlotProperties></PlotProperties>
</ELCOMVariable>
```

name is the ELCOM keyword for the input/output variable.

refNumber is an ELCOM BC reference number. This will override any value given in a **ELCOMBC** configuration element.

title is a display title. If not specified ARMS has a default map of ELCOM Variables to display names.

units is the units of the variable. If not specified ARMS has a default map of ELCOM Variables to units.

5.3.13 ELCOM Bathymetry File Resources

ARMS contains an additional resource factory for creating resources that link to ELCOM Bathymetry files. The schema for the configuration file is simply

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ELCOMBathymetry>
  <fileName></fileName>
</ELCOMBathymetry>
```

where **fileName** is the relative path of the bathymetry file.

5.4 ARMS Configuration

The previous sections have discussed the configuration of the major types of ARMS resources. The resource are placed into an ARMS instance using the main ARMS Configuration file (nominally *arms.xml*). The schema for the main configuration is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ARMSConfig>
  <name></name>
  <longname></longname>
  <latitude>0.0</latitude>
  <longitude>0.0</longitude>
  <timeZone></timeZone>
  <levelLinkerFile>config/services/levellinker.xml</levelLinkerFile>    <timeLineFile>config/services/timeline.xm
  <colorLimitsFile></colorLimitsFile>
  <ARMSContextConfig>
    <address></address>
    <resourceType></resourceType>
    <fileName></fileName>
  </ARMSContextConfig>
</ARMSConfig>
```

Where

name is a reference name for the configuration

longname is a display name for the configuration

latitude, **longitude** and **timeZone** are the default values for the configuration which will be used if there is no value specified for a sensor or station.

levelLinkerFile is the relative location of the Level Linker configuration file. The level linker allows for conversion between different height datums (see Section 5.5).

timeLineFile is the relative location of the Time Line file (see Section 5.6).

colorLimitsFile is one or more relative locations of files that can be used to override color limits for all resources with a given name.

The **ARMSContextConfig** sub-elements are used to call the different resource factories described above.

address gives the ARMS address for the base resource. These are dot separated strings of the form *station-group.stationname*

resourceType gives the type of resource to be generated and should be one of 1. Station 2. ELCOM 3. ELCOM Bathymetry File

fileName is the relative location of the resource configuration file.

5.5 Level Linker File

The level linker allows for conversion of heights between different height datums. The XML file has the following schema:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LevelLinker>
  <comment></comment>
  <mainDatum></mainDatum>
  <Datum>
    <name></name>
    <title></title>
    <type></type>
    <address></address>
    <scaleFactor>1.0</scaleFactor>
    <offset>0.0</offset>
  </Datum>
</LevelLinker>
```

Where

MainDatum: A string giving the name of the primary datum for this site. This should be a recognised standard datum such at the Australian Height Datum.

Datum: The datum sub-elements provide a series of datum heights which ARMS can convert between. Datums may be either static of dynamic (changing in time). Each sub element has the following elements.

name: A keyword name of the datum for referencing from ARMS tasks and plots.

title: A title for display purposes.

type: The type of the datum must be one of "static" or "dynamic".

address: The address of the ARMS resource to be used for the basis of the dynamic datum. ARMS searches for resources near the current resource with a name equal to address. So if we are setting the height of a resource "lds.t1.downsampled.thermistor.thermistor1" and the address is given as "surface" then ARMS searches for resources named one of "lds.t1.downsampled.thermistor.surface", "lds.t1.downampled.surface", "lds.t1.surface" and "lds.surface". If more than one resource is found the deepest resource is used. If no resource if found an error will be given.

scaleFactor: Valid only for dynamic datums. Gives the multiplication factor for converting from the dynamic datum resource.

offset: A vertical offset (positive upwards) for the datum. For static datums this is the offset from the main datum. For dynamic datums this is the offset from the dynamic datum resource.

5.6 Time Line File

The time line file is primarily created and modified by ARMS but can be manually edited if there are other dates that are of interest to the user. The XML file has the following schema:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TimeLineFile>
  <comment></comment>
  <TimeLine>
    <name></name>
    <date></date>
  </TimeLine>
</TimeLineFile>
```

Thus the time line file consists of any number of **TimeLine** sub-elements. Where

name: Is a reference string for the TimeLine object.

date: Is the time and date of the time line object in ISO format. YYYY-MM-DD hh:mm:ss

Tasks

6.1 Introduction

Data processing in ARMS is under taken by performing various tasks on files and data stored in the database. Tasks are available for performing tasks such as obtaining data from a FTP server, importing data, quality checking, down-sampling, averaging and generating model files.

Help information for some of the major tasks carried out in a ARMS is available through the application help menu.

6.2 Task List Files

Tasks are not generally run individually but are run as part of a **tasklist** file. Task list files control what tasks are to be run and in what order they will be run.

Task list files can be run from the tasks GUI menu, from the Configuration file tree in the GUI or can be run using the *cwr.arms.tasks.TaskList* class of the ARMS jar file. The command for running an ARMS task file from the command line is:

```
java -cp /path/to/arms.jar cwr.arms.tasks.TaskList /path/to/arms.xml /path/to/taskfile
```

The schema of the TaskList xml file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TaskListConfig>
  <comment></comment>
  <taskRootDir></taskRootDir>
  <skipNTasks></skipNTasks>
  <TaskConfig>
    <taskName></taskName>
    <configFile></configFile>
    <continueOnError>false</continueOnError>
  </TaskConfig>
  <taskListFile></taskListFile>
</TaskListConfig>
```

Where

taskRootDir This is simply a relative path (from the main ARMS folder) to a directory that will become the base directory for all task files listed below. This parameter is optional and if not specified then all $configFile$ references below should be to the main ARMS workspace.

TaskConfig (required, multiple allowed) The TaskConfig sub-elements provide the main building blocks for setting up which tasks will be run. Each TaskConfig sub-element specifies the task to be run and the

control file for the task. Each TaskConfig XML element therefore has the following sub-elements.

taskName is the keyword name of the task

configFile points to the relative path of the control file. The path is relative to the *taskRootDir* if supplied in this *TaskList* file or to the main ARMS folder if *taskRootDir* is not listed. If multiple *configFile* are specified within one *TaskConfig* element then separate tasks are run using the different files in the order given

continueOnError an optional parameter set to true—false to control if processing of this *TaskList* file should continue if an error occurred during this task.

taskListFile Any task list file can recursively list other task list files if desired. This allows the user to separate appropriate tasks into separate files but then have one task file to run the full set of tasks.

6.3 Time Series Processing Tasks

The majority of data processing task that ARMS performed are task that inherit functionality from a base timeseries processing task. They therefore all share some similarity in there control files. All configurations will have a *TimeSeries* sub element with the schema:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PSTSCreateNaN>
  <comment></comment>
  <TimeSeries>
    <sourceAddress></sourceAddress>
    <destinationAddress></destinationAddress>
    <sensorsFile></sensorsFile>
    <startDate></startDate>
    <endDate></endDate>
    <timeZone></timeZone>
    <processingTimestep></processingTimestep>
    <processingOffset>0</processingOffset>
  </TimeSeries>
</PSTSCreateNaN>
```

Where

sourceAddress is the ARMS address of the data to be processed. If this is a folder then all elements contained in the folder that are listed in the *sensorsFile* will be processed.

destinationAddress is the destination ARMS address. This is only required for some Time Series Processing task as some tasks only modify the source resource.

sensorsFile Multiple resources can be operated on simultaneously by specifying a *sensorsFile*. The file is relative to the main ARMS folder. The sensor files lists a series of resources.

startDate is the start date for the timeseries processing. This can be a link to the timeline resource (eg: T4-DATAPROCESSING-START@arms.services.timeline) or a date in ISO format (YYYY-MM-DD hh:mm:ss)

endDate is the end date for the timeseries processing. This can be a link to the timeline resource (eg: T4-DATAPROCESSING-END@arms.services.timeline) or a date in ISO format (YYYY-MM-DD hh:mm:ss)

processingTimestep is the timestep for the processing. This controls the time step (in seconds) that will be used for the processing. Although this is a required element it is not used for some tasks as they use the time step of the source address instead.

processingOffset an optional offset from midnight for processing.

Tasklist Scheduler

7.1 Introduction

ARMS has an the ability to schedule the running of tasklist at regular intervals. This can be useful for automating data processing or for automatically generating images and/or movies and emailing files to key stakeholders or transferring files to a webserver for general access. The scheduler can be setup to either be run as part of the GUI or can be called via the command line using:

```
java -cp /path/to/arms.jar cwr.arms.tasks.Scheduler /path/to/arms.xml
```

This will use the scheduler configuration file defined in the database configuration file.

The schema for scheduler configuration file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TaskSchedulerList>
  <uniqueId>0</uniqueId>
  <emailAddress></emailAddress>
  <TaskScheduler>
    <taskName></taskName>
    <menuName></menuName>
    <taskListFile></taskListFile>
    <TaskStartDateTime>
      <startYear></startYear>
      <startMonth></startMonth>
      <startDay></startDay>
      <startHour></startHour>
      <startMinute></startMinute>
      <startSecond></startSecond>
    </TaskStartDateTime>
    <period>0</period>
    <periodTimeUnit></periodTimeUnit>
    <emailAddress></emailAddress>
    <emailWarningIntervalInHours>0.0</emailWarningIntervalInHours>
  </TaskScheduler>
</TaskSchedulerList>
```

Where

uniqueId is any integer and is used to ensure that only one instance of the scheduler is running at anytime.

emailAddress are one or more optional email addressed that will be emailed for all **TaskScheduler** when they have not completed successfully for a defined period.

The **TaskSchedulerList** then contains one or more **TaskScheduler** sub-elements. Each sub-element points to one **taskListFile**.

The starting time of the scheduler is specified using the **TaskStartDateTime** any missing parameters in this element are set to 0.

The interval at which the task list is run is defined by the **period** and **periodTimeUnit**. **periodTimeUnit** must be one of SECONDS, MINUTES, HOURS or DAYS.

The **TaskScheduler** can be configured to send emails if the Task List has not completed successfully in a given period. The period is given in hours in the **emailWarningIntervalInHours** element. Emails are sent to all **emailAddress** given in the root element and the **TaskScheduler** element.

CHAPTER 8

Plotting

8.1 Introduction

This section outline the XML file format for generating custom plots of one or more ARMS resources.

Three different plot tasks are currently available 'CustomPlot', 'ProfileLinePlot' and 'MoviePlot'. CustomPlot tasks are used to show static line and contour plots of 1D, 2D and Profile resources for a set period of time. ProfileLinePlot tasks show profile data as a function of depth. MoviePlot tasks are used to show the progression of ELCOM simulations over time.

The easiest way to generate plot configuration files is to create an image interactively using the ARMS GUI an then using the Save Plot XML Configuration option in the File Menu.

8.2 Custom Plots

CustomPlot tasks are used to show line and contour plots of 1D, 2D and Profile resources as a function of time. Thus the *x-axis* for Custom Plots is always date. The schema of the CustomPlot xml file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CustomPlot>
  <startDate></startDate>
  <endDate></endDate>
  <title></title>
  <closeFigure>false</closeFigure>
  <refreshPlotInterval>0</refreshPlotInterval>
  <plot>
    <address></address>
    <Data>
      <date></date>
      <value></value>
    </Data>
    <axes></axes>
    <PlotProperties></PlotProperties>
  </plot>
  <SaveToFile>
    <fileName></fileName>
  </SaveToFile>
</CustomPlot>
```

Where

startDate is the start date for the time series plot. This can be a link to the timeline resource (eg: T4-DATAPROCESSING-START@timeline), a reference to the current date such as 'now-2', or a date in ISO format (YYYY-MM-DD hh:mm:ss)

endDate is the end date for the time series plot. This can be a link to the timeline resource (eg: T4-DATAPROCESSING-START@timeline), a reference to the current date such as 'now' or a date in ISO

format (YYYY-MM-DD hh:mm:ss)

title is a string to appear at the top of the plot

closeFigure set to true/false is the figure is to be closed upon generation. This is primarily intended for use with the automatic generation of image files. If set to true the plot will be created and saved without opening on the screen.

refreshPlotInterval an integer refresh rate. If greater than zero the plot will be refreshed every n seconds by re-querying the database looking for new data. This allows the user to leave a plot open on the desktop which will automatically sync with the database.

plot the plot sub-elements provide the main building blocks for setting up the desired image. Each plot element will define how one resource is to be plotted on the image.

address is the address of the resource to be plotted.

The **Data** sub-elements provide a mechanism for plotting arbitrary data on an axes. Each **Data** element has a **date** and **value** element and ARMS will simply plot a line between these points. To use the **Data** sub-element the **address** must be set to the keyword 'data'. For example to plot a horizontal line for the last 7 days the XML would be

```
<address>data<address>
<Data>
  <date>now-7</date>
  <value>10</value>
</Data>
<Data>
  <date>now</date>
  <value>10</value>
</Data>
```

axes is the axes to plot this resource in. The axes number start at 1 from the top of the image.

The **PlotProperties** sub-element provides a number of ways to configure the axes being plotted. See the **PlotProperties** section below for more information.

SaveToFile The SaveToFile sub-element allows the plot to be automatically saved to an image file. The value of the fileName element is a string giving the filename relative to the main ARMS folder

8.2.1 Example

A simple xml file for plotting two thermistors on the same axes is shown below

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CustomPlot>
```

```
  <comment>Testing custom plots</comment>
  <startDate>T4-DATAPROCESSING-START@timeline</startDate>
  <endDate>T4-DATAPROCESSING-END@timeline</endDate>
  <plot>
    <address>lake.lds.t4.downsampled.15min.thermal.tchn01</address>
    <axes>1</axes>
    <PlotProperties></PlotProperties>
  </plot>
  <plot>
    <address>lake.lds.t4.downsampled.15min.thermal.tchn02</address>
    <axes>1</axes>
    <PlotProperties></PlotProperties>
  </plot>
  <SaveToFile>test.png</SaveToFile>
</CustomPlot>
```

8.3 Profile Line Plots

ProfileLinePlot tasks are used to show profile data as a line of value versus depth. The *y-axis* for Custom Plots is always depth below the surface or height above some datum. The schema of the ProfileLinePlot xml file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProfileLinePlot>
  <title></title>
  <closeFigure>false</closeFigure>
  <refreshPlotInterval>0</refreshPlotInterval>
  <profilePlot>
    <date></date>
    <address></address>
    <axes></axes>
    <title></title>
    <titleProperties></titleProperties>
    <PlotProperties><PlotProperties>
  </profilePlot>
  <SaveToFile>
    <fileName></fileName>
  </SaveToFile>
</ProfileLinePlot>
```

title is a string to appear at the top of the plot

closeFigure set to true/false is the figure is to be closed upon generation. This is primarily intended for use with the automatic generation of image files. If set to true the plot will be created and saved without opening on the screen.

refreshPlotInterval an integer refresh rate. If greater than zero the plot will be refreshed every n seconds by re-querying the database looking for new data. This allows you to leave a plot open on the desktop which will automatically sync with the database.

profilePlot the profilePlot sub-elements provide the main building blocks for setting up the desired image. Each plot element will define how one resource is to be plotted on the image.

address is the address of the resource to be plotted. This should point to a profile resource.

The **date** is the date of the profile we wish to plot.

axes is the axes to plot this resource in. The axes number start at 1 from the left of the image.

title is a string to override the default title

titleProperties is a string containing Matlab property value pairs. eg: 'fontsize',22,'color','k'.

The **PlotProperties** sub-element provides a number of ways to configure the axes being plotted. See the **PlotProperties** section below for more information.

SaveToFile The SaveToFile sub-element allows the plot to be automatically saved to an image file. The value of the **fileName** element is a string giving the filename relative to the main ARMS folder

8.4 Movie Plots

Movie plot xml files allow the user to visualise movies of ELCOM simulations. The movie plot will generate a plot of one or more ELCOM sheets or curtains and allow the user to step through in simulation time. The timestep of the movie plot is determined by the timestep of the first plot axes. The schema of the MoviePlot xml file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MoviePlot>
  <startDate></startDate>
  <endDate></endDate>
  <figureWidth></figureWidth>
  <figureHeight></figureHeight>
  <PlotText>
    <text></text>
    <xLocation>0.0</xLocation>
    <yLocation>0.0</yLocation>
    <zLocation>0.0</zLocation>
    <textProperties></textProperties>
  </PlotText>
  <DateText>
    <xDateLocation></xDateLocation>
    <yDateLocation></yDateLocation>
    <textProperties></textProperties>
  </DateText>
  <closeFigure>false</closeFigure>
  <MovieAxes>
    <address></address>
    <axesLocation></axesLocation>
    <hideAxes>false</hideAxes>
    <title></title>
    <titleProperties></titleProperties>
    <PlotText>
      <text></text>
      <xLocation>0.0</xLocation>
      <yLocation>0.0</yLocation>
      <zLocation>0.0</zLocation>
```

```
      <textProperties></textProperties>
    </PlotText>
    <PlotArrow>
      <xLocation>0.0</xLocation>
      <yLocation>0.0</yLocation>
      <angle>0.0</angle>
      <lineColor></lineColor>
      <lineStyle></lineStyle>
      <lineWidth></lineWidth>
      <color></color>
      <width></width>
      <length></length>
    </PlotArrow>
    <PlotProperties></PlotProperties>
  </MovieAxes>
  <SaveToMovie>
    <fileName></fileName>
    <useDateStamp>true</useDateStamp>
    <framerate>10</framerate>
    <useFrames>true</useFrames>
    <interval>1</interval>
    <frameStart>-1</frameStart>
    <frameEnd>-1</frameEnd>
    <overWrite>false</overWrite>
    <compressMovie>false</compressMovie>
    <videoCodec></videoCodec>
  </SaveToMovie>
  <SaveMonthlyArchiveMovie>
    <fileName></fileName>
    <framerate>10</framerate>
    <useFrames>true</useFrames>
    <interval>1</interval>
    <compressMovie>false</compressMovie>
    <videoCodec></videoCodec>
  </SaveMonthlyArchiveMovie>
  <SaveFrames>
    <fileName></fileName>
    <useDateStamp>true</useDateStamp>
    <interval>1</interval>
    <overWrite>false</overWrite>
    <transparentBackground>false</transparentBackground>
  </SaveFrames>
</MoviePlot>
```

startDate is the start date for the movie. This can be a link to the timeline resource (eg: T4-DATAPROCESSING-START@timeline), a reference to the current date such as 'now-2', or a date in ISO format (YYYY-MM-DD hh:mm:ss)

endDate is the end date for the movie. This can be a link to the timeline resource (eg: T4-DATAPROCESSING-START@timeline), a reference to the current date such as 'now' or a date in ISO format (YYYY-MM-DD hh:mm:ss)

figureWidth and **figureHeight** set the size of Movie plots in pixels.

PlotText The PlotText sub-element allows users to display any text on the image. The PlotText XML element has the following sub-elements.

text (required) The string to display.

xLocation (required) The x location of the string in normalised plot coordinates-ordinates. 0 is the left edge of the plot and 1 is the right edge.

yLocation (required) The y location of the string in normalised plot coordinates-ordinates. 0 is the bottom edge of the plot and 1 is the top edge.

textProperties A string containing matlab property value pairs. eg: 'fontsize',22,'color','k'.

8.4.1 DateText

The DateText sub-element allows users to display the date on the image. The DateText XML element is the same as the PlotText element but with no text value.

8.4.2 MovieAxes

The MovieAxes sub-elements provide the main building blocks for setting up the desired image. Each MovieAxes element will define how one resource is to be plotted on the figure. Each MovieAxes XML element has the following sub-elements.

address The address of the resource to be plotted. Movies can be generated from ELCOM sheets and curtains but also from standard time series resources.

axesLocation The position of the axes to plot this resource in. If this is a four element array of the form [**x y w h**] then a new axes is created with the position bottom left hand corner at x and y and a width and height of w, h. All coordinates are normalised so a **axesLocation** of [**.2 .1 .5 .3**] will create an axes which has a bottom left corner at 20% of the width of the figure and 10% of the height and has a width of 50% and height of 30% of the figure. If the axes location is an integer then the resource is plotted on an existing axes with a value of 1 indicating it should be plotted on the 1st axes created.

hideAxes set to true to hide the axes for this plot. Only the data is shown.

title is a string to override the default title

titleProperties is a string containing matlab property value pairs. eg: 'fontsize',22,'color','k'.

when **PlotText** sub-elements are specified within a MovieAxes element then the x and y locations are in axes coordinate

The **PlotArrow** sub-element allows users to display an arrow on the image. The **PlotArrow** XML element has the following sub-elements.

xLocation and **yLocation** give the starting point of the arrow in plot coordinates.

angle is the angle of the arrow in degrees clockwise from positive x.

lineColor and set **color** the colour of the arrow line and head respectively. The string can be one of the Matlab color keys *b, g, y, c, m, r, k , w* or a Matlab RGB triple such as [.5 .5 .5] where the color component values are between 0 and 1.

lineStyle can be one of '-' (solid), '–' (dashed), ':' dotted or '-.' dash-dot.

lineWidth sets the width of the arrow line

width sets the width of the arrow head

length is the length of the arrow in pixels.

The **PlotProperties** sub-element provides a number of ways to configure the axes being plotted. See the plotProperties section below for more information.

SaveToMovie The SaveToMovie sub-element allows the plot to be automatically saved to a movie file.

fileName gives the the filename relative to the main ARMS folder. This should be one of **.avi* or **.mov*.

if **useDateStamp** is set to true the file name will be modified to include the date range of the movie.

framerate set the frame rate of the movie in frames per second.

interval set to an integer to only include every n^{th} frame.

frameStart and **frameEnd** allow the user to save only a portion of the movie.

overWrite set to true to automatically over write any existing file at the save location.

compressMovie set to true to compress the movie using **ffmpeg** after saving. The location of the external program **ffmpeg** needs to be included in the users preferences.

videoCodec set the compression codec to use with **ffmpeg**. Valid values are h264, mpeg4 and msmpeg4v2.

the SaveMonthlyArchiveMovie element is essentially the same as **SaveToMovie** element but ARMS will automatically break the movie up into 1 movie for each moth of simulation time.

SaveFrames saves individual frames as png images. If the **transparentBackground** element is set to true the images will have a transparent background.

8.5 Plot Properties

The **PlotProperties** element contains a large number of optional elements that can be used to control how resources are plotted. **PlotProperties** elements are (in order of precedence) part of Plot configuration files, sensor information and station resources. For all plotting ARMS first looks at the **PlotProperties** in the plot file, if any element is missing the sensor and station resources are then checked. If any element is not specified in any of the configuration then the default value is used. Due to the length of the XML schema in will be broken into sections for discussion

8.5.1 Layout

```
<Layout>
  <axesHeightScale></axesHeightScale>
  <xTop></xTop>
  <yRight></yRight>
  <northAngle></northAngle>
  <xOffset></xOffset>
  <yOffset></yOffset>
  <zOffset></zOffset>
  <leftMargin>70</leftMargin>
  <rightMargin>50</rightMargin>
  <topMargin>50</topMargin>
  <bottomMargin>50</bottomMargin>
</Layout>
```

axesHeightScale is used in time series plots to change the height of a particular axes. Normally all axes will have the same height but this allows one axes to be scaled differently.

xTop For profile line plots set to true to use the top X axes for this plot. This allows multiple resource to be plotted on the one plot with different x scales

yRight Set to true to use the right Y axes for this plot. This allows multiple resource to be plotted on the one plot with different y scales

xOffset, **yOffset** and **zOffset** are used in 3-Dimensional movie plots and allow a particular resource to be offset in any direction from it's actual location.

leftMargin, **rightMargin**, **topMargin** and **bottomMargin** control the amount of space surrounding an axis.

8.5.2 Text

```
<Text>
  <legendName></legendName>
  <xLabel></xLabel>
```

```
  <yLabel></yLabel>
  <DateText>
    <xDateLocation></xDateLocation>
    <yDateLocation></yDateLocation>
    <textProperties></textProperties>
  </DateText>
  <PlotText>
    <text></text>
    <xLocation>0.0</xLocation>
    <yLocation>0.0</yLocation>
    <zLocation>0.0</zLocation>
    <textProperties></textProperties>
  </PlotText>
</Text>
```

legendName is a string to override the default value that appears in the legend

xLabel is a string to override the default x axis label

yLabel is a string to override the default y axis label

DateText is a sub-element that allows the current date to be drawn on a movie axes. **xDateLocation** and **yDateLocation** are the in the axes coordinates. **textProperties** is a Matlab string formatting statement such as 'fontsize',12

8.5.3 Colors

```
<Colors>
  <colorMap></colorMap>
  <lineColor></lineColor>
  <lineStyle></lineStyle>
  <currentDateColor></currentDateColor>
  <bathColor></bathColor>
  <meshColor></meshColor>
</Colors>
```

colorMap The colormap to use for color plots one of jet, flip_jet, gray, flip_gray, hot, flip_hot, cool, flip_cool, autumn, flip_autumn, spring, flip_spring, summer, flip_summer, winter, flip_winter. The flip prefix reverses the direction of the colormap.

lineColor overrides the default line color for a line plot. The string can be one of the Matlab color keys *b, g, y, c, m, r, k , w* or a Matlab RGB triple such as [.5 .5 .5] where the color component values are between 0 and 1.

lineStyle can be one of '-' (solid), '–' (dashed), ':' dotted or '-.' dash-dot.

currentDateColor overrides the default line color showing the current date in time series movies axes. The string format is the same as for **lineColor**.

bathColor Sets the color of bathymetry in Matlab curtain plots.

meshColor Sets the color of grid meshes in Matlab plots.

8.5.4 ColorBar

```
<ColorBar>
  <colorBarAxesLocation></colorBarAxesLocation>
  <colorBarAxesProperties></colorBarAxesProperties>
  <colorBarType></colorBarType>
  <colorBarLabel></colorBarLabel>
</ColorBar>
```

colorBarAxisLocation The axes location of color bar legend. This is only used in Matlab movie plotting. Default plotting uses the **colorBarType** property.

colorBarAxesProperties Matlab axes properties to be applied to the color bar when using Matlab plotting.

colorBarType An integer that sets the orientation of the colorbar.

- 1: Horizontal at bottom
- 2: Horizontal at top
- 3: Vertical at left
- 4: Vertical at right

colorBarLabel A string to replace the default label of the colorbar.

8.5.5 Limits

```
<Limits>
  <xLimits></xLimits>
  <yLimits></yLimits>
  <zLimits></zLimits>
  <datum></datum>
  <dateAxes></dateAxes>
  <movingDateAxes></movingDateAxes>
  <dateFormat></dateFormat>
  <logVar1></logVar1>
  <minDepthLimit></minDepthLimit>
  <colorLimits></colorLimits>
  <contourLevels></contourLevels>
  <nanMinClim></nanMinClim>
  <nanMaxClim></nanMaxClim>
</Limits>
```

xLimits Set this to a string of the form $[x1\ x2]$ to limit the x axes.

yLimits Set this to a string of the form [$y1$ $y2$] to limit the y axes.

zLimits Set this to a string of the form [$z1$ $z2$] to limit the z axes.

datum modify y values in a depth dependent time series plot by changing heights to use the given level linker datum.

dateAxes true/false to show/hide date labels.

movingDateAxes is used for time series movie plots. By using this option the axes will scroll from right to left as the date increases. Set this to a string of the form [$a1$ $a2$]. The x axis limits are then set to the current date - a1 days : current date + a2 days.

dateFormat the format of the date labels. For example 'yyyy-MM-dd hh:mm'.

logVar1 Set to 1 to plot the log value of the resource. Currently only available with Matlab plotting

colorLimits Set this to a string of the form [$c1$ $c2$] to limit the color axes of contour plots.

contourLevels Set this to a string of the form [$a : b : c$] to set the contour levels. a is the minimum contour level, b is the contour interval and c is the maximum contour level. This is currently only used for Matlab plotting.

nanMinClim and **nanMaxClim** can be set to 1 to hide data outside color limits in Matlab plotting.

8.5.6 NaNRegions

```
<NaNRegions>
  <nanRegion></nanRegion>
  <nanRegionZ></nanRegionZ>
  <heightNaNFill></heightNaNFill>
  <surfaceNaNFill></surfaceNaNFill>
</NaNRegions>
```

nanRegion To set a specified region of a color 2D and 3D movie plot to NaN. This can be useful to allow views of curtains through the top sheet. Format is [$x1$ $x2$ $y1$ $y2$] all values of the plot in the region defined by $x1 < x < x2$ and $y1 < y < y2$ are set to NaN. Only used in Matlab plotting.

heightNaNFill Used only for ELCOM 3D sheets. Allows setting of NaN values in the sheets height to a uniform value. Only used in Matlab plotting.

surfaceNaNFill Used only for ELCOM 3D sheets. Allows setting of NaN values in the sheets value to a uniform value. Only used in Matlab plotting.

8.5.7 PlotType

```
<PlotType>
  <plotType></plotType>
  <shadeType></shadeType>
  <plot3D></plot3D>
  <plotProfileDelta></plotProfileDelta>
  <plotHeight></plotHeight>
  <plotDepth></plotDepth>
  <plotStdDevPeriod></plotStdDevPeriod>
  <plotMeanPeriod></plotMeanPeriod>
  <plotResolution></plotResolution>
  <showLine></showLine>
  <showShape></showShape>
  <shapeStyle></shapeStyle>
  <fillShape></fillShape>
  <barPlot></barPlot>
</PlotType>
```

plotType The type of plot used for color 2D and 3D movie plots in Matlab.

shadeType The type of shading used for color 2D and 3D movie plots in Matlab.

plot3D For ELCOM plots set this to true to plot sheets and curtains using the full 3-D information. Currently only available using the Matlab plotting

plotProfileDelta is for profile time series resources. Set this to true to plot a line showing the difference between the surface and bottom values.

plotHeight is for profile time series resources. Set this to a number to plot the value at a particular height relative to the main datum.

plotDepth is for profile time series resources. Set this to a number to plot the value at a particular depth below the surface level.

plotStdDevPeriod is for time series resources. Set this to a number to plot a rolling standard deviation of the data using the given window in days.

plotMeanPeriod is for time series resources. Set this to a number to plot a rolling mean of the data using the given window in days.

showLine, **showShap**, **shapeStyle**, **fillShape** and **barPlot** control the look of time series line plots. Data can be displayed as line plots with or without shapes at the data points of as a bar plots. Valid **shapeStyle** values are s (square), o (circle) + (cross), ^ (triangle pointing up), d (diamond) and v (triangle pointing down).

8.5.8 QuiverPlot

```
<QuiverPlot>
  <showQuiver>false</showQuiver>
  <scale></scale>
  <nx></nx>
  <ny></ny>
  <lineColor></lineColor>
  <legendLocation></legendLocation>
  <legendVelocity></legendVelocity>
</QuiverPlot>
```

The **QuiverPlot** element controls the plotting of velocity arrows on 2-Dimensional ELCOM sheet movies.

Set **showQuiver** to true to plot the velocity arrows.

scale controls the size of the velocity arrows. The value of the **scale** is the size of a 1 m/s velocity in metres in the x and y grid scale.

nx and **ny** are integer values used to subsample the velocity field. For example to plot the velocities only every third grid in both directions set both **nx** and **ny** to 3.

legendLocation is a two element vector of the form $[x\ y]$ giving the x and y location (in grid coordinates) of the start of a legend arrow.

legendLocation is a two element vector of the form $[u\ v]$ giving the u and v velocities of the legend arrow.

8.5.9 WindPlot

```
<WindPlot>
  <windVectorColor></windVectorColor>
  <windCircleColor></windCircleColor>
  <windCircles></windCircles>
  <scaleFactor>1.0</scaleFactor>
</WindPlot>
```

8.5.10 Matlab

```
<Matlab>
  <axesProperties></axesProperties>
  <matlabCommands></matlabCommands>
</Matlab>
```

axesProperties Allows user to specify any Matlab axes properties in the plot using any valid Matlab property value pair. String argument is passed directly through to a $set(gca, ...)$ call.

matlabCommmands Allows user to specify any valid Matlab command to be run after the plot is complete.

String argument is passed directly through $eval(...)$ call.

Aquarius Time-Series Connection

ARMS is capable of connecting to *Aquatic Informatics™ Aquarius Time-Series* data software. Connection is done via the Aquarius REST API. This section outlines the XML file format of the required XML connection file and details how resources for the Aquarius sensors can be created. The file must be called *ARMSAquariusConnections.xml* and must be located in the current ARMS Workspace. The file should be created by the ARMS Administrator and distributed to the appropriate users.

9.1 ARMSAquariusConnections.xml

The *ARMSAquariusConnections.xml* file specifies the details for connecting to the REST API. The XML file has the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<AquariusConnectionList>
  <AquariusRestConnection>
    <name>AQUARIUS_REST</name>
    <url>http://aquarius.server.com/AQUARIUS/Publish/AquariusPublishRestService.svc</url>
    <PasswordLookup>AQUARIUS_REST</PasswordLookup>
  </AquariusRestConnection>
</AquariusConnectionList>
```

where

name A name for this connection.

url A valid HTTP URL pointing to the location of the REST service.

passwordLookup The Key in *ARMSPasswordsRepository.xml* that holds the username/password information.

9.2 Aquarius Resources

Once the Aquariust Connection information has been specified through the *ARMSAquariusConnections.xml* file ARMS will create a root node in the GUI resource tree. This root node gives access to all sensor data available. Due to the time is can take to load all sensors this node is not populated until the user requests the resources by refreshing the node (right click, *Refresh*).

Resources pointing to individual Aquarius sensors or to specific Aquarius locations can also be added to any other ARMS Resource tree. Individual sensors can be added to ARMS Station Resources by including them in the appropriate sensor file (see Section 5.2.6). An example sensor file for connection to two meteorological sensors is:

```
<?xml version="1.0" encoding="UTF-8"?>
<AquariusConnectionList>
<SensorFile>
  <SensorAquariusRest1D>
    <name>air_pressure</name>
    <dataID>PA.Verified@415800</dataID>
    <serviceName>AQUARIUS_REST</serviceName>
    <SensorInfo>
      <title>Air Pressure</title>
      <units>kPa</units>
    </SensorInfo>
  </SensorAquariusRest1D>
  <SensorAquariusRest1D>
    <name>precipitation</name>
    <dataID>PP.Verified@415800</dataID>
    <serviceName>AQUARIUS_REST</serviceName>
    <SensorInfo>
      <title>Precipitation</title>
      <units>m</units>
    </SensorInfo>
  </SensorAquariusRest1D>
</SensorFile>
```

where

dataID is the Aquarius sensor ID.

serviceName is the name of the connection given in the *ARMSAquariusConnections* file.

To create an Aquarius Location ID the *Aquarius Location* resource type is used in the *ARMSConfig* file. An Aquarius Location Resource is a specialised ARMS station resource where all information (latitude, longitude and sensors) is created from the information contained on the Aquarius server. An example configuration element is:

```
    <address>aquarius.met</address>
    <resourceType>Aquarius Location</resourceType>
    <fileName>/config/resources/met/415800/aquarius.xml</fileName>
  </ARMSContextConfig>
```

an example configuration file is:

```
<?xml version="1.0" encoding="UTF-8"?>
<AquariusLocation>
  <locationId>415800</locationId>
  <serviceName>AQUARIUS_REST</serviceName>
</AquariusLocation>
```

where

locationId is the Aquarius Location ID.

serviceName is the name of the connection given in the *ARMSAquariusConnections* file.

Case Study

10.1 Lake Diagnostic System

In this section we will outline all the XML configuration files required for a simple ARMS configuration with one Lake Diagnostic System Resource. We discuss both the resource and task list configuration files.

10.1.1 Sensors

The example LDS will have the following sensors.

- Wind Speed and Direction
- 2 Thermistors floating at the surface at depths of 1 and 4 metres
- 2 Thermistors attached to the bottom at heights of 1 and 3 metres
- A depth sensor located 0.5 metre off the bottom

To describe the sensors three sensor configuration files are setup.

These files will be located in the **site/config/resources/lds/sensors/** directory.

wind.xml describes the wind sensors and derived wind vector and looks like

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SensorFile>
  <Sensor1D>
    <name>windspeed</name>
    <SensorInfo>
      <FileInfo>
        <channel>WSPD</channel>
      </FileInfo>
      <Location>
        <datum>LDS_SURFACE</datum>
        <zOffset>1.7</zOffset>
      </Location>
      <Limits>
        <minValue>0.01</minValue>
        <maxValue>100.0</maxValue>
      </Limits>
      <Defaults>
        <defaultValue>0.0</defaultValue>
      </Defaults>
    </SensorInfo>
  </Sensor1D>
  <Sensor1D>
    <name>winddir</name>
    <SensorInfo>
      <FileInfo>
        <channel>WDIR</channel>
      </FileInfo>
      <Location>
        <datum>LDS_SURFACE</datum>
```

```
        <zOffset>1.7</zOffset>
      </Location>
      <Limits>
        <minValue>0.0</minValue>
        <maxValue>360.0</maxValue>
      </Limits>
      <Defaults>
        <defaultValue>0.0</defaultValue>
      </Defaults>
    </SensorInfo>
  </Sensor1D>
  <Sensor2D>
    <name>windvector</name>
    <SensorInfo>
      <Location>
        <datum>LDS_SURFACE</datum>
        <zOffset>1.7</zOffset>
      </Location>
      <Limits>
        <minXValue>0</minXValue>
        <maxXValue>100</maxXValue>
        <minYValue>0</minYValue>
        <maxYValue>100</maxYValue>
      </Limits>
      <Defaults>
        <defaultXValue>0</defaultXValue>
        <defaultYValue>0</defaultYValue>
      </Defaults>
    </SensorInfo>
  </Sensor2D>
</SensorFile>
```

The file has two **Sensor1D** elements which point to one data stream in the LDS Files. The **channel** elements gives the LDS channel name. The **Sensor2D** element is a vector resource derived from the wind speed and direction. The vector resource is primarily created to get around problems with down-sampling and averaging of direction data.

To simplify configuration ARMS has a list of common keywords name which are automatically converted into titles and units. For example the Sensor with the name **winddir** is automatically given the title **Wind Direction** and units **deg**. A list of all the available keywords and there corresponding titles and units is available from the ARMS help menu.

The **Location** sub-element defines the position of the sensors relative to a datum. The meteorological sensors are all attached to a floating buoy and are thus given the **datum** LDS_SURFACE (the datum must be specified in the LevelLinker file) and an offset of 1.7 metres. When they heights are set for this data ARMS will look for the surface resource and set the heights of these sensors to 1.7 metres above it at the given times.

The **Limits** sub-element set the limits to be used for the minimum and maximum quality checking. The **defaultValue** is used when using the default filling method.

thermistors.xml defines the sensor information for the thermistor chain. The file has the following content

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SensorFile>
  <Sensor1D>
    <name>thermistor1</name>
    <SensorInfo>
      <FileInfo>
        <channel>TCHN 1</channel>
      </FileInfo>
      <Location>
        <datum>LDS_SURFACE</datum>
        <zOffset>-1.0</zOffset>
      </Location>
    </SensorInfo>
  </Sensor1D>
  <Sensor1D>
    <name>thermistor2</name>
    <SensorInfo>
      <FileInfo>
        <channel>TCHN 2</channel>
      </FileInfo>
      <Location>
        <datum>LDS_SURFACE</datum>
        <zOffset>-4.0</zOffset>
      </Location>
    </SensorInfo>
  </Sensor1D>
  <Sensor1D>
    <name>thermistor2</name>
    <SensorInfo>
      <FileInfo>
        <channel>TCHN 3</channel>
      </FileInfo>
      <Location>
        <datum>LDS_BOTTOM</datum>
        <zOffset>+3.0</zOffset>
      </Location>
    </SensorInfo>
  </Sensor1D>
  <Sensor1D>
    <name>thermistor4</name>
    <SensorInfo>
      <FileInfo>
        <channel>TCHN 4</channel>
      </FileInfo>
      <Location>
        <datum>LDS_BOTTOM</datum>
        <zOffset>+1.0</zOffset>
      </Location>
    </SensorInfo>
  </Sensor1D>
</SensorFile>
```

The **Sensor1D** elements are essentially the same as for the meteorological sensors. **thermistor3** and **thermistor4** are attached to the bottom and thus are given a datum of **LDS_BOTTOM**.

No defaults or limits are given for the sensors as these will be specified for the profile group as a whole in the station configuration file.

depth.xml configuration files defines a sensor for the raw depth sensor and another sensor for specifying the level of the water surface relative to the main datum. The **depth.xml** file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SensorFile>
  <Sensor1D>
    <name>depth</name>
    <SensorInfo>
      <FileInfo>
        <channel>DEPTH</channel>
      </FileInfo>
      <Location>
        <datum>LDS_BOTTOM</datum>
        <zOffset>0.5</zOffset>
      </Location>
      <Limits>
        <minValue>0</minValue>
        <maxValue>100</maxValue>
      </Limits>
      <Defaults>
        <defaultValue>12</defaultValue>
      </Defaults>
    </SensorInfo>
  </Sensor1D>
  <Sensor1D>
    <name>surface</name>
    <SensorInfo>
      <Location>
        <datum>AHD</datum>
        <zOffset>0</zOffset>
      </Location>
    </SensorInfo>
  </Sensor1D>
</SensorFile>
```

Note that as the surface data is generated from the depth data no channel information is required.

10.1.2 Station Configuration

To associate all the LDS sensors into one coherent resource we create a station resource configuration file 'site/config/resources/lds/lds.xml'. The station resource gives the location of the LDS and defines the 'Resource Groups'. Each Resource Groups contains all the same sensors and are used to define how the data will be processed. The station configuration file is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Station>
  <name>LDS</name>
  <title>LDS</title>
  <longname>LDS</longname>
  <site>Demo</site>
  <latitude>-32.0</latitude>
  <longitude>137.0</longitude>
  <timeZone>+8.0</timeZone>
  <daylightSavingsAware>false</daylightSavingsAware>
  <surfaceDatum>LDS_SURFACE</surfaceDatum>
```

```
  <bottomDatum>LDS_BOTTOM</bottomDatum>
  <datum>LDS_SURFACE</datum>

  <StationResourceGroup>
    <address>raw</address>
    <longname>Raw Data</longname>
    <processingInterval>30</processingInterval>
    <processingOffset>0</processingOffset>
  </StationResourceGroup>
  <StationResourceGroup>
    <address>downsampled.15min</address>
    <longname>Downsampled 15 min</longname>
    <processingInterval>900</processingInterval>
    <processingOffset>0</processingOffset>
  </StationResourceGroup>
  <StationResourceGroup>
    <address>averaged.60min</address>
    <longname>Average 60 Min</longname>
    <processingInterval>3600</processingInterval>
    <processingOffset>0</processingOffset>
  </StationResourceGroup>

  <StationSensorsGroup>
    <name></name>
    <sensorsFile>config/resources/lds/sensors/depth.xml</sensorsFile>
  </StationSensorsGroup>
  <StationSensorsGroup>
    <name>met</name>
    <sensorsFile>config/resources/lds/sensors/wind.xml</sensorsFile>
  </StationSensorsGroup>
  <StationProfileGroup>
    <name>thermistor</name>
    <sensorsFile>config/resources/lds/sensors/thermistors.xml</sensorsFile>
    <SensorInfo>
      <Limits>
        <minValue>0.0</minValue>
        <maxValue>30.0</maxValue>
      </Limits>
      <Defaults>
        <defaultValue>15.0</defaultValue>
      </Defaults>
    </SensorInfo>
  </StationProfileGroup>
</Station>
```

Three **StationResourceGroup** are given one for raw data, one for 15 minute downsampled data and one for hourly average data. The limits and default value for all thermistors are given in the **StationProfileGroup** element.

10.1.3 Site Configuration

The site configuration is done in the **arms.xml** file. The file creates one station resource using the **lds.xml** shown above. The file also needs to point to a timeline file and a levellinker file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ARMSConfig>
  <name>site</name>
```

```
  <longname>ARMS</longname>
  <latitude>-32.0</latitude>
  <longitude>100.0</longitude>
  <timeZone>+8.0</timeZone>
  <levelLinkerFile>config/services/levellinker.xml</levelLinkerFile>
  <timeLineFile>config/services/timeline.xml</timeLineFile>

  <ARMSContextConfig>
    <address>lds</address>
    <resourceType>Station</resourceType>
    <fileName>config/resources/lds/lds.xml</fileName>
  </ARMSContextConfig>
</ARMSConfig>
```

Once all the resource files have been created and the database has been initialised the resource tree for the simple LDS is created as shown in Figure 10.1.

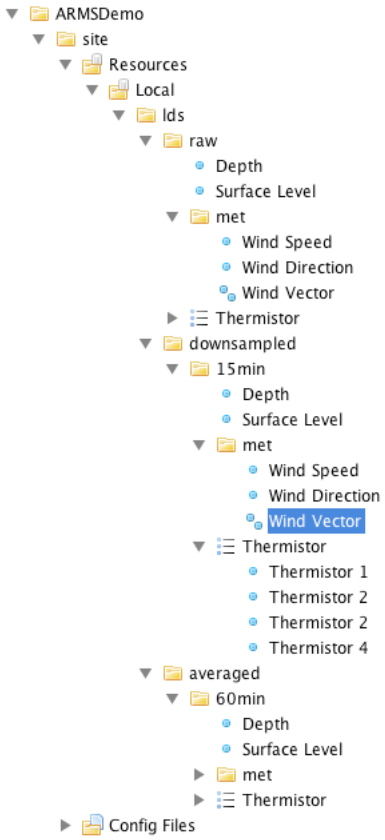

Figure 10.1 The Resource tree for a simple LDS

10.1.4 Data Processing

Processing of a Lake Diagnostic System usually undertakes the following steps:

1. Downloading Raw data files
2. Importing Raw data from text files
3. Setting processing time period

4. Creating surface level resource from the depth sensor resource
5. Quality Checking data
6. Create Wind Vector Resource
7. Set the height of the data points
8. Downsample data
9. Fill missing data
10. Average data

These steps are shown in the following task list configuration file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TaskListConfig>
  <taskRootDir>config/tasks/lds/</taskRootDir>
  <TaskConfig>
    <taskName>FTP Files From Server</taskName>
    <configFile>raw/ftpsync.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>LDS Import</taskName>
    <configFile>raw/import.xml</configFile>
  </TaskConfig>

  <TaskConfig>
    <taskName>Timeline Update From List</taskName>
    <configFile>timeline/start_dataprocessing.xml</configFile>
    <configFile>timeline/end_dataprocessing.xml</configFile>
  </TaskConfig>

  <TaskConfig>
    <taskName>Time Series Quality Checker</taskName>
    <configFile>qualitycheck/depth_qualitycheck.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Time Series Set Height</taskName>
    <configFile>raw/set_depth_height.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Create Surface</taskName>
    <configFile>raw/create_surface_height.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Time Series Downsampler</taskName>
    <configFile>downsampled/surface_downsample.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Time Series Filler</taskName>
    <configFile>filled/surface_fill.xml</configFile>
  </TaskConfig>

  <TaskConfig>
    <taskName>Time Series Quality Checker</taskName>
    <configFile>qualitycheck/wind_qualitycheck.xml</configFile>
    <configFile>qualitycheck/thermal_qualitycheck.xml</configFile>
  </TaskConfig>

  <TaskConfig>
    <taskName>Vector Creator</taskName>
```

```
    <configFile>raw/createWindVector.xml</configFile>
  </TaskConfig>

  <TaskConfig>
    <taskName>Time Series Set Height</taskName>
    <configFile>raw/set_thermistor_heights.xml</configFile>
    <configFile>raw/set_wind_heights.xml</configFile>
  </TaskConfig>

  <TaskConfig>
    <taskName>Time Series Downsampler</taskName>
    <configFile>downsampled/wind_downsample.xml</configFile>
    <configFile>downsampled/thermal_downsample.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Vector Downsampler</taskName>
    <configFile>downsampled/wind_downsample.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Time Series Set Height</taskName>
    <configFile>downsampled/set_thermistor_heights.xml</configFile>
    <configFile>downsampled/set_wind_heights.xml</configFile>
  </TaskConfig>

  <TaskConfig>
    <taskName>Time Series Filler</taskName>
    <configFile>filled/wind_fill.xml</configFile>
    <configFile>filled/thermal_fill.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Vector Filler</taskName>
    <configFile>filled/wind_fill.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Time Series Set Height</taskName>
    <configFile>filled/set_thermistor_heights.xml</configFile>
    <configFile>filled/set_wind_heights.xml</configFile>
  </TaskConfig>

  <TaskConfig>
    <taskName>Time Series Averager</taskName>
    <configFile>averaged_60min/surface_average.xml</configFile>
    <configFile>averaged_60min/wind_average.xml</configFile>
    <configFile>averaged_60min/thermal_average.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Vector Averager</taskName>
    <configFile>averaged_60min/wind_average.xml</configFile>
  </TaskConfig>
  <TaskConfig>
    <taskName>Time Series Set Height</taskName>
    <configFile>averaged_60min/set_thermistor_heights.xml</configFile>
    <configFile>averaged_60min/set_wind_heights.xml</configFile>
  </TaskConfig>
</TaskListConfig>
```

For information on the various tasks see the ARMS help menu.