Argo FileChecker

Installation and User Instructions

Software Version: 2.5

Table of Contents

[1 Document History 3](#_Toc415147733)

[2 Introduction 3](#_Toc415147734)

[3 Install and Build 3](#_Toc415147735)

[3.1. Install from “executable distribution” 4](#_Toc415147736)

[3.2. Install from “source distribution” 4](#_Toc415147737)

[4 Usage 4](#_Toc415147738)

[4.1. Executing the FileChecker 4](#_Toc415147739)

[4.2. FileChecker Results Files – XML option 6](#_Toc415147740)

[4.2.1 Basic Results – XML Option 6](#_Toc415147741)

[4.2.2 Meta-data Results Files – XML Option 7](#_Toc415147742)

[4.2.3 Profile Results File – XML Option 8](#_Toc415147743)

[4.2.4 Technical Results File – XML Option 10](#_Toc415147744)

[4.2.5 Trajectory Results File – XML Option 11](#_Toc415147745)

[4.3. FileChecker Results Files – Text option 12](#_Toc415147746)

[4.3.1 Basic Results File – Text option 12](#_Toc415147747)

[4.3.2 Meta-data Results Files – Text Option 13](#_Toc415147748)

[4.3.3 Profile Results File – XML Option 14](#_Toc415147749)

[4.3.4 Technical Results File – Text Option 16](#_Toc415147750)

[4.3.5 Trajectory Results File – Text Option 17](#_Toc415147751)

[5 Specification Files 18](#_Toc415147752)

[5.1. Reference Tables 18](#_Toc415147753)

[5.2. Meta-data Configuration Parameters File 19](#_Toc415147754)

[5.3. Technical Parameters Files 21](#_Toc415147755)

[5.4. Physical Parameter Files 23](#_Toc415147756)

[Appendix A. Creating the distribution 24](#_Toc415147757)

[Appendix B. File Manifest 25](#_Toc415147760)

# Document History

|  |  |
| --- | --- |
| Date | Description |
| 2014-12-05 | * NetCDF libraries updated to v4.3   + Previous version (v4.0) dated back to 2009   + Latest version (v4.5) requires Java v7. Running v6 on US GDAC. * Added “-list-file” option * Updated command line to reflect new arguments |
| 2015-02-10 | * Document new results file entry: “phase” * Document process for generating the argo-physical-parameter spec file * Document new “format-only” options * Add “Creating the distribution” appendix |
| 2015-05-01 | * Update reference file description (now allows blank and comment lines) * Add description of the CDL, OPT, and attr\_regexp files * Add “version info” that is now the results file |
| 2015-07-07 | * Add process for downloading info from Google docs |
| 2015-11-12 | * Update “Install and Build” for new file names, etc |
| 2016-04-12 | * Remove “-no-fresh” option; Add “-no-name-check” option * Document “WARN” feature in \*attr\_regex files |

# Introduction

The Argo FileChecker is implemented in Java. It is distributed as a single “jar” file with all of required components included. In addition to the “FileChecker” source files developed at the US-GDAC, the following third party support libraries are included in the distribution:

* UNIDATA NetCDF-Java (<http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/>). Currently running version 4.3.

This is the “core” package to create/read/write netCDF files.

* XMLBuilder. Version 1.1

This library is used as to allow simple generation of XML file check results.

* Simple Logging Facade for Java (SLF4J). Currently using version 1.7.7.

This library provides the “logging facility” used within the FileChecker. Extensive logging has been included for both informational and debugging purposes.

The system has been tested with Java versions 1.7 and version 1.8. The FileChecker \*.class files that are distributed have been built with jdk\_1.6.?????

## Changes from last revision

Significant changes from the last revision are highlighted as shown: Changes to tests. Editorial changes are not highlighted.

# Install and Build

The software is distributed in two forms: 1) the “executable distribution”, and 2) the “source distribution”. The executable distribution is simply that: a directly executable version of the system where no “build” step is required. The “source distribution” is the source code for the system.

The specification files are required for both of the above distributions.

## Install from “executable distribution”

File name: argo\_filechecker\_exec\_*yyyy-mm-dd*.tar (where *yyyy-mm-dd* is the release date)

Make a directory to hold the Argo File Check system and extract the distribution file into that location using tar. (For example: cd *<install-dir>*; tar –xvf file\_checker.tar) The extracted files are all of the files needed to run the FileChecker.

## Install from “source distribution”

File name: argo\_filechecker\_src\_*yyyy-mm-dd*.tar (where *yyyy-mm-dd* is the release date)

Make a directory to hold the Argo File Check system and extract the distribution file into that location using jar. (For example: cd *<install-dir>*; jar –xvf *<install-file>*  .)

Set the CLASSPATH variable as follows:

*<install-dir>*:*<install-dir>*/logging/log4j-1.2.15.jar:*<install-dir>*/logging/slf4j-api-1.7.7.jar:*<install-dir>*/logging/slf4j-log4j12-1.7.7.jar:*<install-dir>*/unidata/netcdf-4.3.jar

Building is pretty simple. The third-party libraries do not need to be rebuilt. The supported build mechanism is GNU “make”. Simply navigate to the bin/java directory and execute make.

## Install “specification files”

File name: argo\_filechecker\_spec\_*yyyy-mm-dd*.tar (where *yyyy-mm-dd* is the release date)

The specification files can be installed anywhere the user desires. One possibility is to install them in the *<install-dir>* referred to above. Download the “spec tar file” and untar it; a directory named “spec” will be created containing the “spec files”. (For example: cd *<install-dir>*; jar –xvf *<spec-file>*)

# Usage

The FileChecker is designed to process files for only one DAC during an execution. Also, all of the input files are assumed to be in a single directory.

Any number files can be processed during an execution. It is **much more efficient** to process multiple files at a time due to the overhead of reading the specification files; the specification files for a given file-type/version are only read once during an execution.

A FileChecker “results file” is produced for every input file processed. The name of the results file is: *<input-file>*.filecheck; where *<input-file>* is the name of the input file.

## Executing the FileChecker

The command line to execute the FileChecker is as follows:

*For “executable distribution”:*

java -jar ValidateSubmit.jar [options] dac-name spec-dir output-dir input-dir [file-names]

*For “source distribution”:*

java ValidateSubmit [options] dac-name spec-dir output-dir input-dir [file-names]

Options:

-help Help -- this message

-no-name-check Turn off file name validation

default: perform file name validation

-null-warn Perform 'nulls-in-string' check (warning)

default: do NOT check for nulls

-text-result Text-formatted results files

default: XML-formatted results files

-list-file <file-path> File containing list of files to process

default: no list-file (see Input Files below)

-format-only Perform ONLY format checks – no data checks

default: Perform format and data checks

-data-check-all Perform data consistency checks on all file versions

Default: Perform data consistency checks only on v3.1 (and later) files

*-format-only-pre3.1 DEPRECATED – now the default.*

*Retained for backwards compatibility*

Perform ONLY format checks on pre-3.1 files

Format and data checks on 3.1 (and beyond) files

Arguments:

dac-name Name of DAC that owns the input files

spec-dir Directory path of specification files

output-dir Directory path where results files will be placed

input-dir Directory path where input files reside

file-names (Optional) List of files names to process (see below)

Input Files:

Input files to process are determined in one of the following ways (priority order):

1) -list-file List of names will be read from <list-file-path>\n"+

2) [file-names] argument Files listed on command-line will be processed\n"+

3) All files in 'input-dir' will be processed\n"+

## FileChecker Results Files – XML option

The default format for the FileChecker results files is XML. A formal DTD has not been developed. The format of the files is given in the following sections.

**NOTE:** If the file fails the basic format check, all of the meta-data from DATE\_UPDATE to the end of the meta-data will be missing.

### Basic Results – XML Option

<FileCheckResults spec\_version=*”*-r*XXX*” filechecker\_version=”-r *YYY*”>

<file>*full-path-input-file*</file>

<status>*status*</status>

<phase>*last-phase*</phase>

<metadata>

<dac>*dac-name*</dac>

<DATA\_TYPE>*file-type*</DATA\_TYPE>

<FORMAT\_VERSION>*n.m*</FORMAT\_VERSION>

<DATE\_UPDATE>*date-string*</DATE\_UPDATE>

………file-type-specific meta-data (see below)………

</metadata>

<errors number="n">

<error>

*Message*

</error>

…

</errors>

<warnings number="m">

<warning>

*Message*

</warning>

…

</warnings>

</FileCheckResults>

***Definitions:***

*status* One of FILE-ACCEPTED, FILE-REJECTED, ERROR. The status of ERROR means there was an error opening the file (not a NetCDF file, etc) or that the netCDF file did not include the variables necessary to determine the Argo file type and version. The <error> element will contain a description of the error.

*Upper-case names* The upper case names refer to the settings of the associated variables in the data file

*<errors>* The errors element contains descriptions of the errors (format and/or consistency) discovered. If there are any errors, the file status will be FILE-REJECTED.

*<warnings>* The warnings element contains descriptions of the warnings encountered. Warnings do not cause file rejection.

### Meta-data Results Files – XML Option

The file-type-specific meta-data

<DATA\_CENTRE>*data-centre-code*</DATA\_CENTRE>

<PLATFORM\_NUMBER>*platform-number*</PLATFORM\_NUMBER>

<WMO\_INST\_TYPE>*wmo-inst-code* </WMO\_INST\_TYPE>

*Example:*

<FileCheckResults>

<file>…/csiro/submit/7900309\_meta.nc</file>

<status>FILE-REJECTED</status>

<metadata>

<dac>csiro</dac>

<DATA\_TYPE>Argo meta-data</DATA\_TYPE>

<FORMAT\_VERSION>3.1 </FORMAT\_VERSION>

<DATE\_UPDATE>20141024003601</DATE\_UPDATE>

<DATA\_CENTRE>CS</DATA\_CENTRE>

<PLATFORM\_NUMBER>7900309 </PLATFORM\_NUMBER>

<WMO\_INST\_TYPE>846 </WMO\_INST\_TYPE>

</metadata>

<errors number="1">

<error>CONFIG\_PARAMETER\_NAME[16]: Invalid name 'CONFIG'</error>

</errors>

<warnings number="0"/>

</FileCheckResults>

<FileCheckResults>

<file>…/csiro/submit/5904896\_meta.nc</file>

<status>FILE-REJECTED</status>

<metadata>

<dac>csiro</dac>

<DATA\_TYPE>Argo meta-data</DATA\_TYPE>

<FORMAT\_VERSION>3.1 </FORMAT\_VERSION>

<DATE\_UPDATE>20141031065523</DATE\_UPDATE>

<DATA\_CENTRE>CS</DATA\_CENTRE>

<PLATFORM\_NUMBER>5904896 </PLATFORM\_NUMBER>

<WMO\_INST\_TYPE>846 </WMO\_INST\_TYPE>

</metadata>

<errors number="3">

<error>SENSOR\_MODEL[1]: 'SBE41CP-2.0' Invalid</error>

<error>SENSOR\_MODEL[2]: 'SBE41CP-2.0' Invalid</error>

<error>CONFIG\_PARAMETER\_NAME[16]: Invalid name 'CONFIG'</error>

</errors>

<warnings number="0"/>

</FileCheckResults>

### Profile Results File – XML Option

*The file-type-specific meta-data included in the results file are:*

<N\_PROF>*n\_prof*</N\_PROF>

<N\_LEVELS>*n\_levels*</N\_LEVELS>

<DATA\_CENTRE>*data-centre-code(s)*</DATA\_CENTRE>

<PLATFORM\_NUMBER>*platform-number(s)*</PLATFORM\_NUMBER>

<WMO\_INST\_TYPE>*wmo-inst-code*(s)</WMO\_INST\_TYPE>

<CYCLE\_NUMBER>*cycle-number(s)*</CYCLE\_NUMBER>

<DATA\_MODE>*data-mode[n]*</DATA\_MODE>

<DIRECTION>*direction[n]*</DIRECTION>

<JULD-dtg> juld-date-string(s) </JULD-dtg>

<LATITUDE> *lat(s)*</LATITUDE>

<LONGITUDE> *lon(s)*</LONGITUDE>

<PROFILE\_TEMP\_QC>*prof-temp-qc[n]*</PROFILE\_TEMP\_QC>

<PROFILE\_PSAL\_QC>*prof-sal-qc[n]*</PROFILE\_PSAL\_QC>

<PROFILE\_DOXY\_QC>*prof-doxy-qc[n]*</PROFILE\_DOXY\_QC>

In the case of multi-profile files (N\_PROF > 1), the meta-data for each profile is given

* For values shown as “value(s)”, the individual values are separated by commas (“,”).
* For values shown as “value[n]”, the individual values are part of a string of single characters for each profile.

*Examples:*

<FileCheckResults>

<file>…/csiro/submit/R5903957\_082.nc</file>

<status>FILE-ACCEPTED</status>

<metadata>

<dac>csiro</dac>

<DATA\_TYPE>Argo profile</DATA\_TYPE>

<FORMAT\_VERSION>3.1 </FORMAT\_VERSION>

<DATE\_UPDATE>20141103163602</DATE\_UPDATE>

<DATA\_CENTRE> CS,CS</DATA\_CENTRE>

<PLATFORM\_NUMBER> 5903957 ,5903957 </PLATFORM\_NUMBER>

<WMO\_INST\_TYPE> 846 ,846 </WMO\_INST\_TYPE>

<N\_PROF>2</N\_PROF>

<N\_LEVELS>992</N\_LEVELS>

<CYCLE\_NUMBER> 082,082</CYCLE\_NUMBER>

<DATA\_MODE>AA</DATA\_MODE>

<DIRECTION>AA</DIRECTION>

<JULD-dtg> 20141103114724,20141103114724</JULD-dtg>

<LATITUDE> -29.2680,-29.2680</LATITUDE>

<LONGITUDE> 102.1680,102.1680</LONGITUDE>

<PROFILE\_TEMP\_QC>AA</PROFILE\_TEMP\_QC>

<PROFILE\_PSAL\_QC>AA</PROFILE\_PSAL\_QC>

<PROFILE\_DOXY\_QC>null</PROFILE\_DOXY\_QC>

</metadata>

<errors number="0"/>

<warnings number="0"/>

</FileCheckResults>

<FileCheckResults>

<file>…/csiro/submit/D5903650\_002.nc</file>

<status>FILE-REJECTED</status>

<metadata>

<dac>csiro</dac>

<DATA\_TYPE>Argo profile</DATA\_TYPE>

<FORMAT\_VERSION>3.1 </FORMAT\_VERSION>

<DATE\_UPDATE>20141103155516</DATE\_UPDATE>

<DATA\_CENTRE> CS</DATA\_CENTRE>

<PLATFORM\_NUMBER> 5903650 </PLATFORM\_NUMBER>

<WMO\_INST\_TYPE> 846 </WMO\_INST\_TYPE>

<N\_PROF>1</N\_PROF>

<N\_LEVELS>992</N\_LEVELS>

<CYCLE\_NUMBER> 002</CYCLE\_NUMBER>

<DATA\_MODE>D</DATA\_MODE>

<DIRECTION>A</DIRECTION>

<JULD-dtg> 20110209220030</JULD-dtg>

<LATITUDE> -16.0960</LATITUDE>

<LONGITUDE> 154.9260</LONGITUDE>

<PROFILE\_TEMP\_QC>A</PROFILE\_TEMP\_QC>

<PROFILE\_PSAL\_QC>A</PROFILE\_PSAL\_QC>

<PROFILE\_DOXY\_QC>null</PROFILE\_DOXY\_QC>

</metadata>

<errors number="1">

<error>D-mode: CALIBRATION\_DATE[1,1,2]: Not set for 'TEMP'</error>

</errors>

<warnings number="0"/>

</FileCheckResults>

### Technical Results File – XML Option

No additional file-type-specific meta-data are in the technical file.

<DATA\_CENTRE>*data-centre-code*</DATA\_CENTRE>

<PLATFORM\_NUMBER>*platform-number*</PLATFORM\_NUMBER>

*Example:*

<FileCheckResults>

<file>…/csiro/submit/./1901152\_tech.nc</file>

<status>FILE-ACCEPTED</status>

<metadata>

<dac>csiro</dac>

<DATA\_TYPE>Argo technical data</DATA\_TYPE>

<FORMAT\_VERSION>3.1 </FORMAT\_VERSION>

<DATE\_UPDATE>20141103103550</DATE\_UPDATE>

<DATA\_CENTRE>CS</DATA\_CENTRE>

<PLATFORM\_NUMBER>1901152 </PLATFORM\_NUMBER>

</metadata>

<errors number="0"/>

<warnings number="0"/>

</FileCheckResults>

### Trajectory Results File – XML Option

The file-type-specific meta-data in the technical file is:

<DATA\_CENTRE>data-centre-code</DATA\_CENTRE>

<PLATFORM\_NUMBER>platform-number</PLATFORM\_NUMBER>

<WMO\_INST\_TYPE>*inst-code(s)*</WMO\_INST\_TYPE>

<DATA\_MODE>*data-mode(s)*</DATA\_MODE>

<min\_latitude>*min-lat*</min\_latitude>

<max\_latitude>*max-lat*</max\_latitude>

<min\_longitude>*min-lon*</min\_longitude>

<max\_longitude>*max-lon*</max\_longitude>

*Example:*

<FileCheckResults>

<file>…/csiro/submit/1901161\_traj.nc</file>

<status>FILE-REJECTED</status>

<metadata>

<dac>csiro</dac>

<DATA\_TYPE>Argo trajectory</DATA\_TYPE>

<FORMAT\_VERSION>2.2 </FORMAT\_VERSION>

<DATE\_UPDATE>20130830014055</DATE\_UPDATE>

<DATA\_CENTRE>CS</DATA\_CENTRE>

<PLATFORM\_NUMBER>1901161 </PLATFORM\_NUMBER>

<WMO\_INST\_TYPE>846 </WMO\_INST\_TYPE>

<DATA\_MODE>RRRRRRR</DATA\_MODE>

<min\_latitude>-14.8720</min\_latitude>

<max\_latitude>-11.1530</max\_latitude>

<min\_longitude>50.6800</min\_longitude>

<max\_longitude>57.6920</max\_longitude>

</metadata>

<errors number="7">

<error>variable: PRES\_ADJUSTED: not defined in data file</error>

<error>variable: PRES\_ADJUSTED\_QC: not defined in data file</error>

<error>variable: PRES\_ADJUSTED\_ERROR: not defined in data file</error>

<error>variable: TEMP\_ADJUSTED: not defined in data file</error>

<error>variable: TEMP\_ADJUSTED\_QC: not defined in data file</error>

<error>variable: TEMP\_ADJUSTED\_ERROR: not defined in data file</error>

<error>Parameter PSAL: Variables are missing for this parameter

Required variables: 'PSAL', 'PSAL\_QC', 'PSAL\_ADJUSTED', 'PSAL\_ADJUSTED\_QC', 'PSAL\_ADJUSTED\_ERROR'

Reported variables: 'PSAL', 'PSAL\_QC'

Missing variables: 'PSAL\_ADJUSTED', 'PSAL\_ADJUSTED\_QC', 'PSAL\_ADJUSTED\_ERROR'</error>

</errors>

<warnings number="0"/>

</FileCheckResults>

## FileChecker Results Files – Text option

The results files can optionally output in a text format; “-t” option on command line.

**NOTE:** If the file fails the basic format check, all of the meta-data from DATE\_UPDATE to the end of the meta-data will be missing.

### Basic Results File – Text option

VERSION-INFO: FileChecker = ‘-r*XXX*’ Specification = ‘*-rYYY’*

FILE-NAME: *full-path-input-file*

STATUS: *status*

PHASE: last-phase

META-DATA: start

DAC: *dac-name*

TYPE: *data-type*

FORMAT\_VERSION: *version*

DATE\_UPDATE: *date-string*

…………file-type-specific meta-data settings…………

META-DATA: end

FORMAT-ERRORS: start

*error-message-1*

*error-message-n*

FORMAT-ERRORS: end

FORMAT-WARNINGS: start

*warning-message-1*

*warning-message-n*

FORMAT-WARNINGS: end

***Definitions:***

*status* One of FILE-ACCEPTED, FILE-REJECTED, ERROR. A status of “ERROR: *message*” means there was an error opening the file (not a NetCDF file, etc) or that the netCDF file did not include the variables necessary to determine the Argo file type and version.

*error-message-n* The error messages contain descriptions of the errors discovered. Multiple messages are separated by blank lines. Multi-line messages are possible (no blank lines between lines of message). If there are any errors, the file status will be FILE-REJECTED.

*<warnings>* The warnings element contains descriptions of the warnings encountered. Warnings do not cause file rejection. (Similar format to error messages.)

### Meta-data Results Files – Text Option

The file-type-specific meta-data

DATA\_CENTRE: *data-centre-code*

PLATFORM\_NUMBER: *platform-number*

WMO\_INST\_TYPE: *wmo-inst-code*

*Example:*

FILE-NAME: /raid0/godae/argo/test/incoming/csiro/submit/7900309\_meta.nc

STATUS: FILE-REJECTED

META-DATA: start

DAC: csiro

TYPE: Argo meta-data

FORMAT\_VERSION: 3.1

DATE\_UPDATE: 20141024003601

DATA\_CENTRE: CS

PLATFORM\_NUMBER: 7900309

WMO\_INST\_TYPE: 846

META-DATA: end

FORMAT-ERRORS: start

CONFIG\_PARAMETER\_NAME[16]: Invalid name 'CONFIG'

FORMAT-ERRORS: end

FORMAT-WARNINGS: start

FORMAT-WARNINGS: end

FILE-NAME: …/csiro/submit/5904896\_meta.nc

STATUS: FILE-REJECTED

META-DATA: start

DAC: csiro

TYPE: Argo meta-data

FORMAT\_VERSION: 3.1

DATE\_UPDATE: 20141031065523

DATA\_CENTRE: CS

PLATFORM\_NUMBER: 5904896

WMO\_INST\_TYPE: 846

META-DATA: end

FORMAT-ERRORS: start

SENSOR\_MODEL[1]: 'SBE41CP-2.0' Invalid

SENSOR\_MODEL[2]: 'SBE41CP-2.0' Invalid

CONFIG\_PARAMETER\_NAME[16]: Invalid name 'CONFIG'

FORMAT-ERRORS: end

FORMAT-WARNINGS: start

FORMAT-WARNINGS: end

### Profile Results File – Text Option

*The file-type-specific meta-data included in the results file are:*

N\_PROF: *n\_prof*

N\_LEVELS: *n\_levels*

DATA\_CENTRE: *data-centre-code(s)*

PLATFORM\_NUMBER: *platform-number(s)*

WMO\_INST\_TYPE: *wmo-inst-code*(s)

CYCLE\_NUMBER: *cycle-number(s)*

DATA\_MODE: *data-mode[n]*

DIRECTION: *direction[n]*

JULD-DTG: juld-date-string(s)

LATITUDE: *lat(s)*

LONGITUDE: *lon(s)*

PROFILE\_TEMP\_QC: *prof-temp-qc[n]*

PROFILE\_PSAL\_QC: *prof-sal-qc[n]*

PROFILE\_DOXY\_QC: *prof-doxy-qc[n]*

In the case of multi-profile files (N\_PROF > 1), the meta-data for each profile is given

* For values shown as “value(s)”, the individual values are separated by commas (“,”).
* For values shown as “value[n]”, the individual values are part of a string of single characters for each profile.

*Examples:*

FILE-NAME: …/csiro/submit/R5903957\_082.nc

STATUS: FILE-ACCEPTED

META-DATA: start

DAC: csiro

TYPE: Argo profile

FORMAT\_VERSION: 3.1

DATE\_UPDATE: 20141103163602

N\_PROF: 2

N\_LEVELS: 992

DATA\_CENTRE: CS,CS

PLATFORM\_NUMBER: 5903957 ,5903957

WMO\_INST\_TYPE: 846 ,846

CYCLE\_NUMBER: 082,082

DATA\_MODE: AA

DIRECTION: AA

JULD-DTG: 20141103114724,20141103114724

LATITUDE: -29.2680,-29.2680

LONGITUDE: 102.1680,102.1680

PROFILE\_TEMP\_QC: AA

PROFILE\_PSAL\_QC: AA

PROFILE\_DOXY\_QC: null

META-DATA: end

FORMAT-ERRORS: start

FORMAT-ERRORS: end

FORMAT-WARNINGS: start

FORMAT-WARNINGS: end

FILE-NAME: …/csiro/submit/D5903650\_002.nc

STATUS: FILE-REJECTED

META-DATA: start

DAC: csiro

TYPE: Argo profile

FORMAT\_VERSION: 3.1

DATE\_UPDATE: 20141103155516

N\_PROF: 1

N\_LEVELS: 992

DATA\_CENTRE: CS

PLATFORM\_NUMBER: 5903650

WMO\_INST\_TYPE: 846

CYCLE\_NUMBER: 002

DATA\_MODE: D

DIRECTION: A

JULD-DTG: 20110209220030

LATITUDE: -16.0960

LONGITUDE: 154.9260

PROFILE\_TEMP\_QC: A

PROFILE\_PSAL\_QC: A

PROFILE\_DOXY\_QC: null

META-DATA: end

FORMAT-ERRORS: start

D-mode: CALIBRATION\_DATE[1,1,2]: Not set for 'TEMP'

FORMAT-ERRORS: end

FORMAT-WARNINGS: start

FORMAT-WARNINGS: end

### Technical Results File – Text Option

No additional file-type-specific meta-data are in the technical file.

DATA\_CENTRE: *data-centre-code*

PLATFORM\_NUMBER: *platform-number*

*Example:*

FILE-NAME: …/csiro/submit/1901152\_tech.nc

STATUS: FILE-ACCEPTED

META-DATA: start

DAC: csiro

TYPE: Argo technical data

FORMAT\_VERSION: 3.1

DATE\_UPDATE: 20141103103550

DATA\_CENTRE: CS

PLATFORM\_NUMBER: 1901152

META-DATA: end

FORMAT-ERRORS: start

FORMAT-ERRORS: end

FORMAT-WARNINGS: start

FORMAT-WARNINGS: end

### Trajectory Results File – Text Option

The file-type-specific meta-data in the technical file is:

DATA\_CENTRE: data-centre-code

PLATFORM\_NUMBER: platform-number

WMO\_INST\_TYPE: *inst-code*

DATA\_MODE: *data-mode(s)*

MIN-LATITUDE: *min-lat*</min\_latitude

MAX-LATITUDE: *max-lat*</max\_latitude

MIN-LONGITUDE: *min-lon*</min\_longitude

MAX-LONGITUDE: *max-lon*</max\_longitude

*Example:*

FILE-NAME: …/csiro/submit/1901161\_traj.nc

STATUS: FILE-REJECTED

META-DATA: start

DAC: csiro

TYPE: Argo trajectory

FORMAT\_VERSION: 2.2

DATE\_UPDATE: 20130830014055

DATA\_CENTRE: CS

PLATFORM\_NUMBER: 1901161

WMO\_INST\_TYPE: 846

DATA\_MODE: RRRRRRRRRRRR

MIN-LATITUDE:-14.8720

MAX-LATITUDE:-11.1530

MIN-LONGITUDE:50.6800

MAX-LONGITUDE:57.6920

META-DATA: end

FORMAT-ERRORS: start

variable: PRES\_ADJUSTED: not defined in data file

variable: PRES\_ADJUSTED\_QC: not defined in data file

variable: PRES\_ADJUSTED\_ERROR: not defined in data file

variable: TEMP\_ADJUSTED: not defined in data file

variable: TEMP\_ADJUSTED\_QC: not defined in data file

variable: TEMP\_ADJUSTED\_ERROR: not defined in data file

Parameter PSAL: Variables are missing for this parameter

Required variables: 'PSAL', 'PSAL\_QC', 'PSAL\_ADJUSTED', 'PSAL\_ADJUSTED\_QC', 'PSAL\_ADJUSTED\_ERROR'

Reported variables: 'PSAL', 'PSAL\_QC'

Missing variables: 'PSAL\_ADJUSTED', 'PSAL\_ADJUSTED\_QC', 'PSAL\_ADJUSTED\_ERROR'

FORMAT-ERRORS: end

FORMAT-WARNINGS: start

FORMAT-WARNINGS: end

# Specification Files

The specification files used by the FileChecker are described in the following sections. The source of the data is documented and the manipulations necessary to prepare the data for use in the FileChecker are described.

All of the specification files contain “version information” in the form of the Subversion (svn) revision number and the date the file was last modified. These are relative to the US GDAC svn repository; I suppose these could someday be hosted on a public code repository like Github, etc.

## Format specification

The allowed format of an Argo netCDF file is defined using (netCDF) CDL file, an optional OPT file, and an optional attribute regular expression file. The nominal file name standard for these files is:

argo-*<file-type>*-spec-v*<version>*.cdl

argo-*<file-type>*-spec-v*<version>*.opt

argo-*<file-type>*-spec-v*<version>*.attr\_regexp

where

* *<file-type>* is the type of Argo file: metadata, profile, technical, trajectory, b\_profile, or b\_trajectory
* *<version>* is the format version number. For instance, “2.3”, “3.1”, …

### CDL File

The “CDL spec file” defines the basic format of the Argo netCDF file. The “CDL spec file” is in the Unidata netCDF CDL file format; the same format that the netCDF “ncdump” utility would generate from a valid Argo netCDF file.

The “CDL spec file” specifies EXACTLY what global attributes,dimensions, variables, and variable attributes are allowed in an Argo netCDF file, including data types, attributes settings, and dimension values.

Exceptions:

* Optional dimensions and variables: Every variable that is allowed must be included in the CDL spec file; physical parameter variables are handled separately, as discussed below. Some variables are optional. Variables are specified as optional in the OPT file described in the next section.
* Variable dimensions: Some dimensions (obviously) vary from file to file. These are indicated in the CDL as “\_unspecified\_”. For example:
  + N\_PROF = \_unspecified\_;
* Extra dimensions: Some physical parameters are allowed to have an “extra dimension”. This is allowed for in the CDL spec file with a line similar to:
  + N\_VALUES\d+ = \_extra\_;
* Alternate dimensions: Variables can be allowed to have “alternate dimensions”: that is, the dimension may be any of the specified list. These are indicated in the CDL as:
  + type variable (alt-dim1**|**alt-dim2**|**…)

Example:

char PREDEPLOYMENT\_CALIB\_COMMENT(N\_PARAM, **STRING1024|STRING4096**);

* the dimension may be either STRING1024 or STRING4096
* Special attribute settings: Three special attribute settings can be used in the specification files (the CDL file and elsewhere)
  + “<->”: the attribute cannot exist is the data file
  + “<\*>”: the attribute is optional and its value is ignored
  + “<+>”: the attribute is REQUIRED but its value is ignored

Example: JULD:resolution = "<+>";

NOTE: Sometimes you will see a value following this special value. That is a “default setting” when a new Argo netCDF file is created and does not afftect the operation of the FileChecker. Example: :institution = "<+>US GDAC" ;

* The global attributes can have a “regular expression” match defined for them inline in the CDL file. The syntax for this is:

:attribute-name = “preferred-value”; /\* REGEX = “allowed-regex” \*/

Example:

:Conventions = "Argo-3.1 CF-1.6" ; /\*REGEX = "Argo-3\..\* +CF-.\*" \*/

* Physical Parameter Variables: The variables for the physical parameters are NOT included in the CDL file. These are built from the Physical Parameter Files described in Section 5.5 below.

### OPT Files

These files simply contain a list of the dimensions and variables that are considered optional.

Example:

***argo-technical-spec-v2.2.opt***

TECHNICAL\_PARAMETER\_NAME

TECHNICAL\_PARAMETER\_VALUE

### Attribute Regular Expression Files

Sometimes it is desirable to allow slight variations in the attributes. This is particularly useful to allow multiple values to be accepted during a “transition period” after a format modification.

These are specified in the “attr\_regexp spec file” using “regular expression” syntax.

Example:

***argo-profile-spec-v3.1.attr\_regexp***

DOXY:valid\_min = 0\.[1-3];

DOXY\_ADJUSTED:valid\_min = 0\.[1-3]; WARN

DOXY\_STD:valid\_min = 0\.[1-3]; NOWARN

These will be match 0.1, 0.2, or 0.3. Notice that the “.” Is escaped, otherwise it matches anything in that position; you know, typical regular expression syntax.

ALSO, there is an optional setting following the regular expression which can be either WARN or NOWARN. If set to “WARN”, the FileChecker will issue a warning whenever this regular expression is used to allow an attribute. If blank or set to NOWARN, a matching attribute is allowed silently (ie, no warning).

Global attributes can be included in this file (especially if warnings are desired) but a global attribute regex can also be included in the CDL as discussed above.

## Reference Tables

The reference tables required to support format checking have been converted to text files that can be used directly by the format checker. The files are:

|  |  |  |
| --- | --- | --- |
| **File Name** | **Title** | **Source** |
| ref\_table-2 | Quality Control Flag | User's Manual |
| ref\_table-2a | Quality Control Scale | User's Manual |
| ref\_table-5 | Location Classes | User’s Manual |
| ref\_table-6 | Data State | User's Manual |
| ref\_table-8 | Instrument Types | User's Manual |
| ref\_table-9 | Positioning System | User's Manual |
| ref\_table-10 | Transmission System | User's Manual |
| ref\_table-19 | JULD\_\*STATUS | User’s Manual |
| ref\_table-20 | Grounded flag | User’s Manual |
| ref\_table-22 | PLATFORM\_FAMILY | http://tinyurl.com/nwpqvp2 |
| ref\_table-23 | PLATFORM\_TYPE | http://tinyurl.com/nwpqvp2 |
| ref\_table-24 | PLATFORM\_MAKER | http://tinyurl.com/nwpqvp2 |
| ref\_table-25 | SENSOR | http://tinyurl.com/nwpqvp2 |
| ref\_table-26 | SENSOR\_MAKER | http://tinyurl.com/nwpqvp2 |
| ref\_table-27 | SENSOR\_MODEL | http://tinyurl.com/nwpqvp2 |

Important characteristics:

* File formats: The files may be multi-column files with “|” as the column separator. ~~There~~ **~~must~~** ~~be at least two columns. If there isn't naturally a second column, e.g. ref\_table-25, simple put “| “ at the end of each line.~~
* ONLY THE FIRST COLUMN is significant. (Seems senseless but “it's the rules”. Someday, there might be a use for the other columns.)
* The google doc files can simply be “downloaded” as tab-separated files, and then replace the tab with the “|”. (Better than comma-separated because some of the info contains commas).
* Blank lines and lines whose first non-space characters are “//” (comment lines) are ignored. (Comment lines are used to include the Subverion (svn) revision number and update date information.)

**Preparation of the Google Docs tables**

1. Open Google Docs in web browser
2. Select tab for desired reference table. (Note that the tab names include the “table name” followed by the table number in “[ ]”.)
3. Select File -> Download as -> Tab separated values. This will download the table onto your local computer. (Note: one of the tabs is actually two tables – table 24 & 26)
4. Edit the file:
   1. Replace tab with “ | ”. The spaces aren’t technically necessary, but make the table easier to read.
   2. Add “//” at the beginning of any header lines
   3. Add “version information” lines to top of file. (Note: See an existing file. Don’t fill in the information since it will automatically be updated by SVN when it is checked into the repository.)

## Meta-data Configuration Parameters File

**File name: argo-config\_names-spec-v<V>, argo-config\_names-spec-v<V>.deprecated**

The meta-data configuration parameters are defined by two Excel files available on the Argo Data Management website (<http://www.argodatamgt.org/Documentation>); [t](http://www.argodatamgt.org/Documentation)he “Configuration parameter names for metadata files, core Argo” and “Configuration parameter names for metadata files, bio-Argo”

These files must be manually pre-processed prior to use in the format checking. Essentially, the relevant columns must be extracted and converted to a text file. The required pre-processing steps are described below.

**Deprecated file:** (this file is optional)

The Excel files *may* contain deprecated configuration parameter names. To allow the FileChecker to accept these names and issue WARNINGS when they are used, the deprecated names are put in the \*.deprecated file; they must NOT be in the main file. When no deprecated names exist, this file will not exist.

The file fomat and characteristics are exactly the same as the main file.

**Preparation of the Configuration Name File:**

1. Download the current files.
2. Open the core-argo file.
3. Convert column A to a text file. (One method: Copy column A (Configuration Variable Name) into a separate “sheet” and save the new sheet as a “comma-separated file (csv)”.)
4. Remove the “units” part of the parameter names from each line in the new file; the units are everything following the last underscore character (“\_”). Any text editor can be used to perform this process.
5. Repeat steps 2-4 for the bio-argo file.
6. Concatenate the two files together.
7. Remove and white-space from <\*> values in the file. It doesn't matter what is between the <>, just that there is no white-space.
8. Add file header. Put “//” comment characters in front of header-lines. See existing file for example.
9. Place the file in the “specification directory” with the name: argo-config\_names-spec-v<V>, where <V> is the technical file version that these parameters correspond to.

Important characteristics:

* A configuration parameter name must be on a line by itself with no embedded white-space.
* Within a name, any sequence of <\*> is will match any number of characters at that position, or none.
* Blank lines and lines that start with (first characters) “//” are ignored.
* Lines that contain any white-space characters *other* than leading and/or trailing white-space are ignored; essentially, pseudo-comment lines.
* Blank lines and lines whose first non-space characters are “//” (comment lines) are ignored. (Comment lines are used to include the Subverion (svn) revision number and update date information.)

## Technical Parameters Files

**File names: argo-tech\_names-spec-v<*V*>, argo-tech\_units-spec-v<*V*>**

There are two optional files to support “transitions” from one name/unit to another:

**argo-tech\_names-spec-v<*V*>.deprecated, argo-tech\_units-spec-v<*V*>.deprecated**

The technical parameters are defined by two Excel files available on the Argo Data Management website (<http://www.argodatamgt.org/Documentation>); the Technical Parameter Names and the Technical Parameter Units.

These files must be manually pre-processed prior to use in the format checking. Essentially, the relevant columns must be extracted and converted to a text file. The required pre-processing steps are described below.

**Preparation of Technical Parameter Names file:**

1. Download and open the current file.
2. Copy column B (parameter codes) into a separate “sheet” and save the new sheet as a “comma-separated file (csv)”.
3. Remove the “units” part of the parameter names from each line in the new file; the units are everything following the last underscore character (“\_”). Any text editor can be used to perform this process.
4. Place the file in the “specification directory” with the name: argo-tech\_names-spec-v<V>, where <V> is the technical file version that these parameters correspond to.

Important characteristics:

* A parameter name must be on a line by itself with no embedded white-space.
* Within a name, any sequence of <\*> will match any number of characters at that position. No embedded white-space within the <\*>!!
* Blank lines and lines that start with (first characters) “//” are ignored.
* Lines that contain any white-space characters *other* than leading and/or trailing white-space are ignored; pseudo comment lines.
* Blank lines and lines whose first non-space characters are “//” (comment lines) are ignored. (Comment lines are used to include the Subverion (svn) revision number and update date information.)

**Preparation of the Technical Parameter Units file:**

1. Download and open the current file.
2. Copy column A (SI version of approved units) into a separate sheet and save the new sheet as a “comma-separated file (csv)”.
3. \*\*\*While the old units are still being accepted\*\*\* Copy column C (old units) into a separate sheet and save this sheet as a separate csv file. This file must be edited to remove the parenthetical unit descriptions from a few of the lines.
4. Place the file(s) in the “specification directory” with the name(s): argo-tech\_names-spec-v<V> and argo-tech\_names-spec-v<V>.old, where <V> is the technical file version that these parameters correspond to.

**DEPRECATED NAMES AND/OR UNITS**: Creating the “\*.deprecated” file (step 4 and 5) will only be required during the transition period from the old names/units to the new names/units. After the transition is complete, these steps can be skipped.

Names/units that match entries in the \*.deprecated file will be accepted but WARNINGS will be issued regarding usage of the deprecated value.

Important characteristics:

* A unit name must be on a line by itself with no embedded white-space.
* Blank lines and lines that start with (first characters) “//” are ignored.
* Lines that contain any white-space characters *other* than leading and/or trailing white-space are ignored.
* Blank lines and lines whose first non-space characters are “//” (comment lines) are ignored. (Comment lines are used to include the Subverion (svn) revision number and update date information.)

## Physical Parameter Files

**File names: argo-physical\_params-spec-v<*V*>**

The physical parameters are defined by an Excel file available on the Argo Data Management website (<http://www.argodatamgt.org/Documentation>); Argo physical parameters list: Core-Argo and B-Argo, *<date>*. The current versions in use by the format checker is documented in the FileChecker specification file.

These files must be manually pre-processed prior to use in the format checking. Essentially, the relevant columns must be extracted and converted to a text file. The required pre-processing steps are described below.

**Preparation of Physical Parameter file:**

1. Download and open the current file.
2. Copy columns B (parameter codes) and D (long\_name) through I (core/bio/intermediate) into a separate “sheet” and save the new sheet as a “tab-separated file (.txt)”. (NOTE: the separator character will be converted in the following step. Tab-separated works better than comma-separated because some of the cells contain commas.)
3. (((((Someday the rest will all be done by running a script. But for now.))))
4. Replace all of the tabs with “ | “.
5. Clean up the “header line”: Add “//” to the front (comment) and clean it up so it makes sense. Remove any extraneous lines at the top and bottom of the file.
6. Replace a “cf standard name” value of “-“ or “ “ (blank) with “<->” (no quotes). (This means: attribute optional, value ignored).
7. Replace a “valid\_min” or “valid\_max” value of “-“ or “ “ (blank) with “<->” (no quotes). (This means: attribute required, value ignored).
8. Add the data type to the first column. Easiest is to add “float | “ to the beginning of every line then manually scan the “comments” column of the Excel file and change the indicated types to “double”.
9. Scan the comments in the Excel file. For any parameter that indicates is can use the N\_VALUESxx “extra” dimension, add a “+” to the end of the data type; for example, “float+ | UV\_IRRADIANCE……”
11. Look at the previous files and copy the header lines. The “Based on” line through the “column header” line. Update the Based on comment so it is current.
12. Place the file in the “specification directory” with the name: argo-tech\_names-spec-v<V>, where <V> is the technical file version that these parameters correspond to.
13. Creating the distribution
    1. Creating the executable distribution
14. In an “export directory”, “svn export” clean copies of the “java” directory (with sub-directories). (This will not produce any of the SVN files.)

* svn export *repository*/bin/java

1. Create a “distribution directory”.
2. Copy the (third party) JAR files to a “distribution directory”.

* From the export directory: find . -name '\*.jar' -exec cp {} *distr-dir* \;

1. Create an XMLBuilder.jar file in the “distribution directory”.

* From *export*/java: jar cf *distr-dir*/XMLBuilder.jar com

1. Compile the java (within the “*export*/java directory”):

* Set CLASSPATH: In *export/*java: . set\_classpath
* Run “make ValidateSubmit.class”

1. Create the log4j.properties file within the “export directory”

* In *export/*java directory: cp –p log4j.properties.info log4j.properties

1. From a new/updated “check out” directory of the current tag:

* Create the “Application.properties” file by running “make”
* Copy the “Application.properties” file to the *export*/java directory

1. Create the ValidateSubmit.jar executable JAR file

* From the *export*/java: make\_exe\_jar *distr-dir*
  + Executes command: jar cfm *distr-dir*/ValidateSubmit.jar MANIFEST.txt log4j.properties Application.properties ValidateSubmit\*.class ResultsFile\*.class usgdac/\*.class

1. Build the distribution TAR file

* From the distribution directory:
  + tar cvf file\_checker\_exec\_yyyy-mm-dd.tar \*.jar
  1. Creating the specification file distribution

1. In an “export directory”, “svn export” clean copies of the “spec”. (This will not produce any of the SVN files.)

* svn export *repository*/bin/spec

This will create the directory “spec” in the export directory

1. Create the VersionInfo.properties file

* In a “check out” directory of *”*/spec”: make
* Copy the VersoinInfo.properties file to the *export*/spec directory.

1. Build the distribution TAR file

* From the export directory:

tar cvf *distr\_dir/*file\_checker\_spec\_yyyy-mm-dd.tar spec

File Manifest

.

An inventory of the system

build.xml

ValidateSubmit.java

usgdac/

usgdac\_src/

usgdac\_doc/

unidata/

netcdfAll-4.3.jar

netcdfAll-4.3.jar.save

README.txt

com/jamesmurty/utils

BaseXMLBuilder.class

NamespaceContextImpl.class

XMLBuilder2.class

XMLBuilder.class

XMLBuilderRuntimeException.class

logging/

log4j-1.2.15.jar

log4j-over-slf4j-1.7.7.jar

slf4j-1.7.7/

slf4j-api-1.7.7.jar

slf4j-ext-1.7.7.jar

slf4j-log4j12-1.7.7.jar

slf4j-migrator-1.7.7.jar

slf4j-simple-1.7.7.jar