Filename Convention for Radiation Belt Storm Probe Common Data Format data files

Original design: R.Freidel. Modifications: R.J.Barnes

Introduction

Although the RBSP mission does not have a requirement for a universal standard format for filenames, it is in the best interests of the mission if the individual instrument teams adopt a consistent filename convention. The instrument teams have agreed to use the Common Data Format (CDF) self-describing data format as the primary format for level 1 and higher RBSP data products. Consequently it has been decided that the filename convention should be based on the existing CDAWEB filename convention for CDF files.

RBSP CDF Filenames

RBSP CDF files are comprised of a number of variable length, alphanumeric fields, followed by a filename suffix (“cdf”) and optional compression (“gz”,”z”,”bz2”). All fields are required, and are delineated by a field separator character, an underscore (“_”). Fields can be further divided into sub-fields, delineated by a dash (“-“). The distinction between a field and a sub-field is that a field is a required element that must always be included in the filename; a sub-field is an optional element that may or may not be present. A filename parser can be safely coded to extract all fields from a filename and can optional further extract sub-fields as needed.

The filename is of the form:

<source>_<type>_<descriptor>_<date>_<version>.cdf.{.bz2,.gz}

Field	Description	Example
<source>	Data source identifier, comprised of sub-fields for mission (“rbsp”), spacecraft (“a” or “b”), and optionally the instrument suite.	“rbsp-a-ect”, “rbsp-b-emfisis”, “rbsp-a”
<type>	Data type, comprised of sub-fields for a short mnemonic data type identifier and an optional integer data release number.	“pre”, “fnl-001”
<descriptor>	A short descriptor of the data included in the file.	“mag-L2”, “rbspice-L3”, “rps-ap003-l3”
<date>	Start date of the file in Universal Coordinated Time (UTC). Dates can either be in the form, “yyyymmdd” or “yyyymmddhhMMss”. (Allow doy format too? – eg. “yyyydoy”)	“20120201”, “20120830103000”
<version>	Version number consisting of the form “X.Y.Z”, where X is the major (interface) number, Y is the minor (quality number) and Z is the revision number.	“v1.1.1”, “v1.2.1”
.cdf.{.bz2, .gz}	Filename suffix, with an optional additional suffix for the compression algorithm applied.	“.cdf”, “.cdf.gz”, “.cdf.bz2”

Notes:

<source>

<type>

The data type identifier is used to specify the providence of the data, for example; preliminary data (“pre”), final data (“fnl”). (Other possibilities would be browse “brw”, burst mode “bst”, etc.)

The release number is an optional field that can be used to group a collection of data products which may have different version numbers. Depending on each instrument teams method of data processing, a file may or may not have a release number. If the release number is omitted, it is assumed to be zero, so that if a team then later decided to use release numbers, this will not cause a subsequent problem in identifying release numbers.

The release number is a monotonically increasing integer that is used to capture a set of data products at a point in the mission defined by the instrument team. Individual data products may have different version numbers, representing different versions of analysis software and calibration, but can have a common release number.

(Clarification needed – for a given release number, are version number of the products locked, eg. A release is a snapshot, you cannot have different version numbers within that release? Version numbers of products can still change but how do you distinguish between the “released” version, and subsequent incremental changes?)

<descriptor>

The descriptor field is a short, human readable description of the data product. It should include the instrument, and the data product level. Finer levels of description down to measurement type and even APID can be used if deemed appropriate.

<date>

The date is specified in Universal Coordinated Time (UTC). The length of the date field defines both the format of the date and also the length of the file. Dates of the form “*yyyymmdd*” represent files that contain one UTC day of data. Files with the longer “*yyyymmddhhMMss*” specification represent files containing one orbit of data.

yyyy	Year
mm	Month
dd	Day
hh	Hour
MM	Minute
ss	Second

<version>

The version number uses a variant of the industry standard version scheme for software of the form “v*X.Y.Z*”

- **X is the interface number**. For a file, this number changes if the actual structure of a file changes (new variables, new digital format, new structure). Software that reads this file would need to be modified. For a code, this number changes if the code now requires differently formatted inputs or produces differently formatted output, or if it would run differently (i.e. 32 v 64 bit architecture).

- **Y is the quality number**. For a file, this number changes if the contents have been updated to be of better quality, e.g. better calibration, better fidelity magnetic field, fixed transmission problems. Nothing else in the file has changed - the same codes as before can read it process it further. For a code, this number changes if an internal processing change has lead to better quality of the data, e.g. an algorithm has been updated for better accuracy, variables changed from single to double precision, etc.

- **Z is the bug fix/revision number**. This number changes to indicate minor bug fixes in either a file or processing code, or indicate the file needed reprocessing. There is no impact on the end user other than recognizing that the quality of this data is better than a lower version and should be used in preference.

Time Conversion and splitting data files:

The filenames for Level 0 PTP files use the mission elapsed day within the filename. The files are generated to match the UTC day as closely as possible, however there will be some discrepancies.

In generating higher level data products, the actual UTC should be found from the contents of the CCSDS telemetry packets and this should be used to generate the correct file for that packet.

(We should have a standard tool that can read any generic PTP level 0 file and spit out the UTC time of the packets).

Parsing Filenames:

Filenames can be parsed by first breaking the filename down into the various fields, and then decoding them. As all fields are required, the extracting fields is a trivial case of string tokenization. In C this can be done using the “strtok” function, in IDL by using “strpos” and in shell scripts using simple pattern expansion operators.

(Write a IDL and C filename decoders).

Filename Ordering:

Release numbers, version, and sub-version numbers do not have leading zeros, this means that a simple alphanumeric sort will not necessarily return the file names in the ascending version order, eg. “V1.9.1” will precede “V.1.10.1” in a file listing. To avoid this problem filenames should be sorted by parsing the filename.

(Provide a simple utility to sort a list of RBSP filenames according to the convention).