DELPHIN 6 Climate Data File Specification, Version 1.0

Technical Report

Andreas Nicolai

Institut für Bauklimatik (Institute of Building Climatology)
Technische Universität Dresden
D-01062 Dresden, Germany
andreas.nicolai@tu-dresden.de

Abstract

This paper describes the file format of the climate data container used by the *DELPHIN*, *THERAKLES* and *NANDRAD* simulation programs. The climate data container format holds a binary representation of annual and continuous climatic data needed for hygrothermal transport and building energy simulation models. The content of the C6B-Format is roughly equivalent to the epw-climate data format.

Contents

1 Introduction

The C6B file format is an extendible binary container format, dedicated to storing climate data. It is meant to be used for annual hourly data, whereby leap years are excluded. Consequently, an annual hourly climate data file will always have 8760 values per climate component. With *climate component* we mean a specific measured quantity such as temperature, relative humidity or air pressure.

The container format can also be used to store measured continuous data of arbitrary sampling frequency or uneven sampling intervals. For this purpose the table with data can be followed by a single data array with the time points. The data format does not use a specific date/time representation, so that daylight-savings time does not need to be addressed. Simulation programs using the data and user interfaces must convert data points to corresponding date/time representations.

The C6B format serves as standard container format for file-based climate databases, used by the simulation software DELPHIN, THERAKLES and NANDRAD of the Institut für Bauklimatik of the Technische Universität Dresden, Germany.

1.1 General File Layout

All C6B files are composed of three sections:

- Magic header section
- Meta data section
- Data section

1.2 Principle Data Types

The file is written using standard data type encoding, listed in Table 1. Strings and arrays are stored in variable-length format. Empty strings and arrays are encoded by a single 32bit unsigned integer value of `0x00000000`. Note that strings *do not* contain a termination character.

Count	Data Type	Description
	Strings	
1	32bit unsigned int	Number of characters in string -> n
n	8bit char	Characters in string
	Array of Doubles	
1	32bit unsigned int	Number of values in array -> n
n	64bit double	Values in array

Table 1: Binary Encoding for Standard Data Types

2 Magic Header and File Version

Data Type	Description
32bit unsigned int	Magic header part 1 (= "CLDF")
32bit unsigned int	Magic header part 2 (= "RLZ!")
32bit unsigned int	Version number
32bit unsigned int	always = 0x00000000

Table 2: Format of Magic Header and Version Encoding

The data file begins with a header structure stored in the first 16 bytes of the file (see Table 2). The first 4 bytes encode the identity of a file, and must be exactly `CLDF` (`0x46444C43` when read as 32bit integer) followed by the next 4 bytes with content `RLZ!` (`0x215A4C52` when read as 32bit integer). This information is used to identify the file type and check that this is indeed a valid C6B file. This also allows disambiguation with other magic file headers and thus guarantees a unique identification in the UNIX world.

The next 8 bytes encode the file format version number consisting of a major and minor number. All minor version changes are downwards compatible. No content sections in the files will be removed or altered when a new minor version is released. Consequently, applications supporting older version of file formats can still read newer files formats. For example, minor versions may add meta information to the file headers that can simply be ignored.

Major version changes indicate significant changes in the file format and result in incompatible files. A major revision number change might require updates for older distributions of the software using C6B files. The maximum major and minor version number is limited to 255, corresponding to an 8 bit number.

2.1 Version Number Encoding

The binary version number is encoded in two 32 bit unsigned int values (64 bit). The actual version numbers are encoded in the first 32 bit unsigned int, which is followed by a dummy 32 bit unsigned int, as shown in the following example (hex).

```
0x010F0000 0x00000000
```

Here, the first byte encodes the major version number (here 1), and the second byte encodes the minor version number (here 15, as 0F in hexadecimal). The remaining 6 bytes of the 8 byte version block are reserved and must be zero.

3 Meta Data Section

Additional meta data describes the content of the data section and provides information about the station where climate data was measured. The meta data section is designed to be easily extensible in future versions. Generally, the meta data section consists of several lines of text with a leading number identifying the number of lines, see Table 3.

Data Type	Description
32bit unsigned int	n - Number of lines/strings to follow
n x String	n times String base data type

Table 3: Meta Data Section

The first two integer values in the header can be used to quickly access the output data content without parsing the header. This is primarily useful when developing automatic data extracting tools that do not require header information. In such cases, development of header parsing functionality is not necessary and with the file offset stored after the magic header, it is possible to seek directory to the data section in the file.

Each of the lines/strings in the header has the format

```
<keyword>=<value>
```

with the following keyword/value pairs currently defined. When reading the file, the keyword and value should be split up at the equal-character '=', and then interpreted according to the definitions in Table 4. The CITY name is mandatory, since it is used for identification. For radiation load calculations, timezone, latitude and longitude are required.

The TIMEZONE parameter must be of integer data type. Longitude is defined positively eastward (e.g. 13.7565 deg for Dresden, Germany). You may add 360 deg to negative longitude values (e.g. -68.9302 = 291.0698 deg for Willemstad, Curaçao), or store the negative value (full value range is from -180..360 deg).

Keyword	Value-Description	Parameter required?
COUNTRY	descriptive name	
CITY	descriptive name	x
WMO	WMO station code	
SOURCE	description of data source	
TIMEZONE	-12..12	x
LATITUDE	in deg, -90..90, always defined northward	x
LONGITUDE	in deg, -180..180, alternatively 0..360	x
STARTYEAR	year where data starts, for non-cyclic data	
ELEVATION	elevation above N.N.	
COMMENT	description about station data/additional information	

Table 4: Keywords and Values in Meta Data Section

4 Data Section

The climatic data is stored as individual value vectors (see Table 5).

Data Type	Description
n x Double array	n times Double array base data type, where n is the number of defined climate data components
Double array	Double array base data type, with time point offsets (optional, only if not using annual hourly data)

Table 5: Meta Data Section

In the current version of the C6B format, exactly 9 climate data arrays have to be stored in order and units[1] given in Table 6 (units given in brackets). That means, that the first data array holds temperatures in degree C, the second holds relative humidity values as percentages (0..100). The last data array holds rain intensity on a horizontal surface in liters per square meter and hour.

Name	Description
Temperature	Ambient temperature [C]
RelativeHumidity	Ambient relative humidity [%, 0..100]
DirectRadiationNormal	Direct short wave radiation in sun's normal direction [W/m2]
DiffuseRadiationHorizontal	Diffuse short wave radiation on horizontal surface [W/m2]
WindDirection	Wind direction [deg], 0 deg - North, 90 deg - East, < 360 deg.
WindVelocity	Wind velocity [m/s]
LongWaveCounterRadiation	Long wave counter radiation [W/m2]
AirPressure	Atmospheric pressure [Pa]
Rain	Rain flux density on horizontal plane [l/m2h] (liters per square meter and hour)

Table 6: Climata Data Components in Data Section

When data for a certain climate data component is not available, the data array should contain only 0 numbers.

4.1 Cyclic Annual Data

Typically, C6B files will hold annual hourly data that can be applied as cyclic data sets. When storing annual hourly data, the vectors/data arrays must have exactly 8760 values. The first data value corresponds to the *end of the first hour* i.e. 1:00:00. Values between points are meant to be linearly interpolated (see Figure 1). The values within the first hour of the year are meant to be interpolated between the last point (day 365, 24:00 is the same as day 0, 0:00) and the value at the end of the first hour as illustrated in Figure 1. Since both values are the same, the data set starts at 1 hour (no redundant storage of value at 0:00:00).

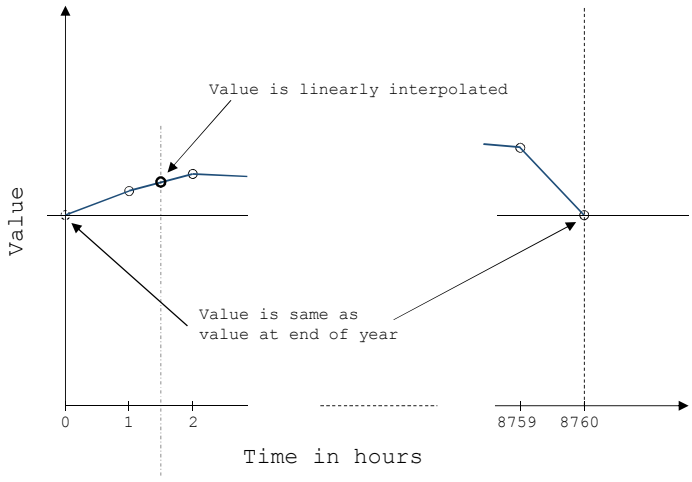

Figure 1: Interpolation of Annual Cyclic Climate Data

[1]The units are selected such that they correspond to typical user's units. Storing data in basic SI units may result in unit conversion errors and numbers like 92 degree might become 91.99999999 degree after conversion.

The arrays with annual data for all climate components are followed by an empty vector (size 0). An empty vector indicates that the data vectors hold hourly values for the whole year.

4.2 Non-Cyclic/Continuous Data

For non-cyclic data, for example from in-situ measurements, the number of values in each data vector can be different from 8760. However, *all vectors must have the same length*. The data is followed by a single data vector that contains the time points corresponding to the data values in the preceding data arrays. If this vector is not empty (length > 0), it must contain *exactly as many values* as all of the data *arrays in the data section*. The time points in the vector are in seconds since begin of the start year as given in the meta data section, see 1. For data that spans several years, the time points may exceed 365*24*3600 seconds. Time points must be strictly monotonically increasing.

The values between the given time points are meant to be linearly interpolated as for the cyclic annual data. Since values before the first time point and past the last time point are not provided, there is no evaluation rule to be applied. Users of continuous climate data must deal with this appropriately (e.g. constant extrapolation or prohibit access to climate data outside the valid date/time range).