SeeYou CUB File Format

The CUB file format is designed to store airspace data that is displayed and utilized by flight navigation software from Naviter and LX Nav.

This format is structured as a binary file comprising three distinct parts: each file begins with a `CubHeader`, followed by a list of `CubItem's`. Each `CubItem` represents a single airspace, detailing its properties and linking to `CubPoint's`, which are cataloged in the final section of the file.

- Format: Binary
- File Extension: `.cub`
- Float byte ordering: Little Endian (LE)
- Integer byte ordering: Little Endian (LE), unless changed in `PcByteOrder`
- Coordinates unit: Radians, positive for `N` and `E`
- String encoding: `UTF-8`, use `Extended ASCII` if string contains incorrect utf-8 sequence

Example implementation of a CUB file parser can be found in our Github Repo: SeeYou File Formats

CubHeader

The `CubHeader` is the initial segment of the CUB file format, spanning the first 210 bytes. It contains metadata about the file and defines how the data should be interpreted, most notably specifying the byte offsets for `CubItem` and `CupPoint` segments.

Bytes	Data Type	Name	Description
4	UINT32	`Ident`	File Identifier, must be `0x425543C2` LE.
112	CHAR[112]	`Title`	Copyright notice.
16	UNIT16[8]	`AllowedSerials`	List of up to 8 device serial numbers authorized to access the file; set to `0` otherwise. Used only when `IsSecured` equals `2`, LE.
1	UNIT8	`PcByteOrder`	Byte ordering flag; BE if set to `0`, LE otherwise.
1	UNIT8	`IsSecured`	Encryption status for data following the header; `0` for no encryption.
4	UINT32	`Crc32`	Reserved for future use (currently ignored).
16	UINT8[16]	`Key`	Encryption key, used if `IsSecured` is not `0`
4	INT32	`SizeOfItem`	Size of single `CubItem`.
4	INT32	`SizeOfPoint`	Size of single `CubPoint`.
4	INT32	`HdrItems`	Number of `CubItem's` contained in the file.
4	INT32	`MaxPts`	Maximal number of points per `CubItem`.
4	FLOAT	`Left`	Left value of data bounding box (longitude).
4	FLOAT	`Top`	Top value of data bounding box (latitude).
4	FLOAT	`Right`	Right value of data bounding box (longitude).
4	FLOAT	`Bottom`	Bottom value of data bounding box (latitude).
4	FLOAT	`MaxWidth`	Maximum width of any `CubItem` (longitude span in radians).
4	FLOAT	`MaxHeight`	Maximum height of any `CubItem` (latitude span in radians).
4	FLOAT	`LoLaScale`	Scaling factor used in shape construction, needed because coordinates are stored as integers. `Coordinate = LoLaScale * Stored Value`
4	INT32	`HeaderOffset`	Byte offset to the first `CubItem`.
4	INT32	`DataOffset`	Byte offset to the first `CubPoint`.
4	INT32	`Alignment`	Reserved for future use (currently ignored).

CubItem

The `CubItem` structure stores essential information about each airspace or NOTAM-defined airspace. Information regarding their storage is detailed in the header. The first `CubItem` is located at the `HeaderOffset` specified in the `CubHeader`, and the file contains a total of `HdrItems` items. Each item occupies `SizeOfItem` bytes, and each subsequent `CubItem` is located immediately following the previous one.

If the `SizeOfItem` is smaller than the total size of the `CubItem` structure (42 bytes), the remaining bytes should be set to 0.

Bytes	Data Type	Name	Description
4	FLOAT	`Left`	Left boundary of the item's bounding box.
4	FLOAT	`Top`	Top boundary of the item's bounding box.
4	FLOAT	`Right`	Right boundary of the item's bounding box.
4	FLOAT	`Bottom`	Bottom boundary of the item's bounding box.
1	UINT8	`Style`	Airspace type; combines highest bit and lowest 4 bits to form `CubStyle`, bits 5-7 represent `CubClass`.
1	UINT8	`AltStyle`	Altitude style for `MinAlt` (lowest 4 bits) and `MaxAlt` (highest 4 bits).
2	INT16	`MinAlt`	Minimum altitude of the airspace (in meters).
2	INT16	`MaxAlt`	Maximum altitude of the airspace (in meters).
4	INT32	`PointsOffset`	Relative byte offset to the first `CubPoint` associated with this airspace from the beginning of the CubPoint's segment.
4	INT32	`TimeOut`	Timeout for this airspace (not used).
4	UINT32	`ExtraData`	Field reserved for additional data.
8	UINT64	`ActiveTime`	Encoded active time affecting this airspace. Set to 0x3FFFFFF as default value if not present

Style Mappings

`CubStyle` categorizes the airspace style.

Value	Description
0x00	Unknown

0x01	Control Zone (CTR)
0x02	Restricted Area (RA)
0x03	Prohibited Area (PA)
0x04	Danger Area (DA)
0x05	Temporary Reserved Area (TRA)
0x06	Terminal Control Area (TMA)
0x07	Traffinc Information Zone/Area (TIZ)
0x08	Airway
0x09	Control Area (CTA)
0x0a	Glider Sector
0x0b	Transponder Mandatory Zone (TMZ)
0x0c	Military Aerodrome Traffic Zone (MATZ)
0x0d	Radio Mandatory Zone (RMZ)
0x0f	NOTAM
0x80	Advisory Aarea
0x81	Air Defence Identification Zone (ADIZ)
0x82	Flight Information Region (FIR)
0x83	Delegated FIR
0x84	Traffic Information Area (TIA)
0x85	Special Rules Zone (SRZ)
0x86	Temporary Flight Restriction (TRA)
0x87	Aerodrome Traffic Zone (ATZ)
0x88	Flight Information Service Area
0x89	Legacy. Maps to RMZ
0x8a	Aerial Sporting and Recreation Area
0x8b	Transponder Recomended Zone (TRZ)
0x8c	VFR Route
0x8d	Alert
0x8e	Temporary Reserved

0x8f	Warning

CubClass Mappings

`CubClass` categorizes the airspace class.

Value	Description
0	Unknown Class
1	Class A
2	Class B
3	Class C
4	Class D
5	Class E
6	Class F
7	Class G

AltStyle Mappings

`AltStyle` specifies the reference used for determining the altitude boundaries (both upper and lower) of an airspace.

Value	Description
0	Unknown
1	AGL (Above Ground Level)
2	MSL (Mean Sea Level)
3	FL (Flight Level)
4	Unlimited
5	Altitude defined by NOTAM

Extra Data

The `Extra Data` field encodes specific NOTAM data when value is not 0 and the highest two bits are set to 0. Below is a breakdown of how this data is structured within the field:

Bits	Description	Values
30-31	Reserved	`00` indicates NOTAM data encoding
28-29	NOTAM Type	`0` : None `1` : Cancel `2` : New `3` : Replace
23-27	First Letter of NOTAM Subject	Encoded as 1 (A) to 26 (Z)
18-22	Last Letter of NOTAM Subject	Encoded as 1 (A) to 26 (Z)
13-17	First Letter of NOTAM Action	Encoded as 1 (A) to 26 (Z)
8-12	Last Letter of NOTAM Action	Encoded as 1 (A) to 26 (Z)
4-6	NOTAM Traffic	`0` : Miscelaneous `1` : IFR `2` : VFR `3` : IFR & VFR `4` : Checklist
0-3	NOTAM Scope	`0` : Unknown `1` : Aerodrome `2` : En-route `3` : Aerodrome and En-route `4` : Nav. Warning `5` : Aerodrome and Nav Warning `8` : Checklist

ActiveTime

Encodes the activation time of an airspace.

Bits	Description
63-52	Days Active Flags.
26-51	Encoded NOTAM Start Date, valid if not 0.
0-25	Encoded NOTAM End Date, valid if not 0x3FFFFFF.

Days Active Flags mapping

Encodes the active days of an airspace.

Flag Value	Description
0x000	Unknown
0x001	Sunday
0x002	Monday
0x004	Tuesday
0x008	Wednesday
0x010	Thursday
0x020	Friday
0x040	Saturday
0x080	Holidays
0x100	AUP (Airspace Use Plan)
0x200	Irregular
0x400	By NOTAM
0x800	Reserved

Unpacking NOTAM Time

NOTAM time is encoded in minutes within the `Active Time` field and is used for both `CubItem ActiveTime` and `CubPoint cdiInserted` type. To unpack this time into a more readable format:

1. **Minutes**: Extract the minutes from the total time using modulo 60. Subtract these minutes from the total time, then divide by 60.
2. **Hours**: Extract the hours from the remaining time using modulo 24. Subtract these hours, then divide by 24.
3. **Days**: Extract the day from the remaining time using modulo 31, add 1 to shift from zero-based to a one-based count, subtract the days, then divide by 31.
4. **Months**: Extract the month from the remaining time using modulo 12, add 1 to shift from zero-based to a one-based count, subtract the months, then divide by 12.
5. **Years**: Add 2000 to the remaining quotient to calculate the year.

This method translates encoded minutes into a full date and time format, allowing for easier interpretation and use of the data.

```
minutes = time%60; time /= 60;
hours = time%24; time /= 24;
days = (time%31) + 1; time /= 31;
months = (time%12) + 1; time /= 12;
years = time+2000;
```

CubPoint

`CubPoint` encodes information about the shape, name, frequency, and other optional attributes of an airspace. The structure is `SizeOfPoint` bytes long, with the first byte serving as a flag that determines how the remaining bytes are interpreted. `SizeOfPoint` will never be less than 5.

1st Byte	2-5th Bytes
Flag	Values, depending on the flag.

Set Origin Offset

Flag `0x81` sets a new origin for subsequent points. Starting origin is `originX = CubItem.Left`, `originY = CubItem.Bottom`. New origin is calculated as `originX += deltaX`, `originY += deltaY`

Bytes	Type	Name	Description
1	UINT8	flag	0x81
2	INT16	x	Set deltaX to (`x * LoLaScale`)
2	INT16	y	Set deltaY to (`y * LoLaScale`)

Add a New Point

Flag `0x01` adds a new point relative to the current origin:

Bytes	Type	Name	Description
1	UINT8	flag	0x01
2	INT16	x	New point X-coordinate (`originX + x * LoLaScale`)
2	INT16	y	New point Y-coordinate (`originY + y * LoLaScale`)

Attribute Record Sequence

Flag with the 7th bit set to 1 indicates the start of a special block of attribute records (flag & 0x40 != 0).

Airspace Name

The lower 6 bits of the first attribute record flag represent the length of the name (up to 63 characters).

Bytes	Type	Name	Description
`SizeOfPoint`	CubPoint	flag	First byte `0x40 + Length`. For example name of length 20 (`0x14`), would result in in flag `0x54`
Defined by first byte	STR	name	Airspace name (max. 63 characters)

Airspace Frequency and Frequency Name

Following the airspace name, the frequency and its associated name may be stored (flag & 0xC0 == 0xC0). The lower 6 bits of the flag represent the length of the name.

Bytes	Type	Name	Description
1	UINT8	flag	`0xC0 + Length`. For example name of length 20 (`0x14`), would result in in flag `0xD4`
4	UINT32	freq	Airspace frequency
Defined by first byte	STR	name	Frequency name (max. 63 characters).

Optional Data

The last optional part of data is interpreted byte by byte based on the second byte named `CubDataId`, which maps to different types of additional data. All the records with flag equal to 0xA0 must be read in a loop (flag == 0xA0).

Bytes	Type	Name	Description
1	UINT8	flag	0xA0
1	UINT8	CubDataId	Determines the type of the following data
1	UINT8	b1	Varies based on `CubDataId`
1	UINT8	b2	Varies based on `CubDataId`
1	UINT8	b3	Varies based on `CubDataId`

Data ID Mappings

Different `CubDataId` values indicate specific types of data that follow the `CubPoint` structure:

Value	Description	Details
0	ICAO Code of the airport	`b3` indicates the length of the following string
1	Secondary Frequency	`b1`, `b2`, `b3` represent the frequency
2	Exception rules to airspace class	`b2` and `b3` define the length of the subsequent string
3	NOTAM Remarks	`b2` and `b3` define the length of the subsequent string
4	NOTAM ID String	`b3` indicates the length of the subsequent string
5	NOTAM Insert date and time	Comprised of `b1`, `b2`, `b3`, and an additional byte after the structure. Read one more byte `b4`. Time is encoded as `(value << 8) + (b4 << 0)`.

Integers composed from multiple bytes are ordered from highest to lowest byte, value = (b1 << 16) + (b2 << 8) + (b3 << 0).