Callsign Lookup Algorithm Specification

Version: 0.30.30-Beta Date: 2025-08-05

--- Revision History ---

[0.30.30-Beta] - 2025-08-05

- No functional changes. Synchronizing version numbers.

[0.30.0-Beta] - 2025-08-05

- Initial release of Version 0.30.0-Beta.

- Standardized all project files to a common baseline version.

---

1. Core Purpose

[cite_start]The script's goal is to replicate the logic of major contest logging programs by implementing a precise, ordered, multi-step algorithm. [cite: 2070] [cite_start]It takes a raw callsign string as input and returns a data tuple containing the resolved DXCC entity, CQ/ITU Zones, continent, and a `portableid` field. [cite: 2071]

2. Output Data Structure

[cite_start]The script's output is a `FullCtyInfo` named tuple, which has been modified to include the `portableid` field. [cite: 2072]

```
(DXCCName, DXCCPfx, CQZone, ITUZone, Continent, Lat, Lon, Tzone, WAEName, WAEPfx, portableid)
```

[cite_start]The `portableid` field will contain the specific part of a portable callsign that was used to determine the location (e.g., "7",

"VP2V") and will be blank for non-portable callsigns. [cite: 2073]

3. The Lookup Algorithm

[cite_start]The script follows a strict order of operations. [cite: 2074] [cite_start]A successful match at any step concludes the algorithm. [cite: 2075]

Step 1: Pre-processing (`_preprocess_callsign`)

[cite_start]The initial step is to clean the raw callsign string to create a standardized base for analysis. [cite: 2076] [cite_start]This involves stripping common non-prefix suffixes such as `/P`, `/M`, `/QRP`, `/B`, and any characters following a hyphen (`-`). [cite: 2076]

Step 2: Exact Match (`_check_exact_match`)

[cite_start]The highest-priority lookup is for an exact match. [cite: 2077] [cite_start]The `CTY.DAT` file can contain entries prefixed with `=` that map a full, unique callsign to a specific entity. [cite: 2078] [cite_start]The script checks for these first. [cite: 2079]

Step 3: Hardcoded Special Cases (`_check_special_cases`)

[cite_start]The script then checks for hardcoded exceptions that do not follow standard patterns. [cite: 2079] The primary rules are:

- [cite_start]Any callsign ending in `/MM` (Maritime Mobile) is immediately classified as an "Unknown" entity. [cite: 2080]
- [cite_start]A specific rule correctly identifies `KG4` callsigns as Guantanamo Bay. [cite: 2081]

Step 4: Portable Call Logic (`_handle_portable_call`)

[cite_start]If the cleaned callsign contains a `/`, it is processed by a dedicated handler that uses a series of heuristics to identify the `portableid`. [cite: 2082] [cite_start]See Section 4 for details. [cite: 2083]

Step 5: Longest Prefix Match (`_find_longest_prefix`)

[cite_start]If the call is not resolved by any of the previous steps, this default lookup method is used. [cite: 2083] [cite_start]It takes the callsign string (e.g., `VP2VMM`) and checks if it is a known prefix. [cite: 2084] [cite_start]If not, it removes the last character and tries again (`VP2VM`), repeating this process until it finds the longest possible valid prefix (`VP2V`) that exists in the `CTY.DAT` data. [cite: 2085]

4. Portable Call Heuristics

[cite_start]The `_handle_portable_call` method uses the following ordered checks. [cite: 2086] [cite_start]If a rule is satisfied, a result is returned and the process stops. [cite: 2087]

1. [cite_start]**Invalid `digit/call` Format:** The script first checks for the invalid `digit/callsign` format (e.g., `7/KD4D`). [cite: 2088] [cite_start]If this pattern is found, the call is considered invalid and returns "Unknown". [cite: 2089]
2. [cite_start]**Unambiguous Prefix Rule:** The script checks if exactly one side of the `/` is a valid prefix in `cty.dat` while the other is not. [cite: 2090] [cite_start]If so, the valid side is identified as the `portableid`. [cite: 2091]
3. [cite_start]**"Strip the Digit" Heuristic:** If the call is still ambiguous, this tie-breaker temporarily strips a trailing digit from each side. [cite: 2092] [cite_start]If this makes one side a valid prefix while the other remains invalid, the original, unmodified side that produced the match is chosen as the `portableid`. [cite: 2093] [cite_start]This is critical for calls like `HC8N/4`. [cite: 2094]
4. [cite_start]**US/Canada Heuristic:** This rule handles the `callsign/digit` format for domestic US/Canada calls. [cite: 2094] [cite_start]If one side appears to have the structure of a US or Canadian callsign and the other is a single digit, the script identifies the **single digit** as the `portableid`. [cite: 2095]
5. [cite_start]**"Ends in a Digit" Heuristic:** This is the final tie-breaker. [cite: 2096] [cite_start]If exactly one side of the `/` ends in a digit while the other does not, the side ending in the digit is identified as the `portableid`. [cite: 2097] [cite_start]This correctly resolves calls like `WT7/OL5Y`. [cite: 2098]
6. [cite_start]**Final Action (No Fallback):** If a call remains ambiguous after all of the above heuristics have been attempted, the script **gives up and returns "Unknown"**. [cite: 2099] [cite_start]The previous "Final Fallback" logic that attempted to guess the location has been removed to prevent incorrect resolutions. [cite: 2099]