Common Code Format for UltraViolet Redemption

Version: 2.10

This document provides important information about the Common Code Format for UltraViolet Code Redemption.

Revision History

Version	Date	Author	Notes
2.9	09.14.15	Laura Poland	Added cover page, revision history, contents and footer. Added Anchor Bay/Starz prefix of “S”. Removed all vowels except “U” from the allowed list of prefixes.
2.10	10/26/2015	Jim Taylor	Much cleanup. Removed repetitively redundant info. Standardized restricted character set for all codes (not just prefixes).

Common Code Format for UltraViolet Redemption

The following code structure allows for identification of the studio from a prefix character without a database lookup.

<u>Required Criteria</u>:

1. A restricted set of characters is used: 2-9, B-D, F-H, J-N, P-T, V-Y
    - "0", "1", "I", and "O" are omitted to avoid confusion
    - "A", "E", and "U" are omitted to avoid creating words
    - (All characters 0-9 are used for the check digit)
2. Studio prefix is required as the first character (unique ID follows)
    - Prefix for studios is selected from the restricted set of characters (with a special exception for "U"):
        - Prefix characters "H", "N", "Z", and "9" are reserved for future use
        - Further restrictions may apply to expand the prefix space (e.g. we may restrict 2-char prefixes such as "ZA" in the future)
    - DECE maintains the registry of studio prefixes
    - Current approved prefixes:

Studio	**Prefix**
Anchor Bay/Starz	S
Fox	4
Lionsgate	C
NBC Universal	U
Paramount	5
Sony Pictures	3
Warner Brothers	7

3. Check digit is required as the last character
    - This is not a security mechanism; it's intended for quick validation in websites, apps, etc.
    - Luhn mod N algorithm is used to compute check digit (see detailed instructions below)
    - Check digit is calculated from the unique ID (part of code between prefix and the check digit)
4. Code length is standardized to 16 or 17, including prefix and check digit
    - Codes with a single-character prefix are 16 characters in length
    - Reserved prefix characters may be used in the future to create variants of the common code with different lengths
5. All codes are required to be case insensitive

<u>Optional Criteria and Best Practices for Generating Codes:</u>

1. No pre-defined format for unique identification, but best practices should be followed:
    - Use uppercase letters
    - Avoid confusable combinations such as "VV" and "WVW"
    - Omission of vowels generally avoids bad words, but the studio may still wish to use a dictionary or word list to eliminate bad words
    - Unique IDs should be randomly generated (never generated in sequence) to eliminate the possibility of guessing codes
    - When printed, codes should be separated into multiple sets divided by space
2. Studios can support additional segmentation of unique ID portion of codes, such as title, offer, or format
    - If additional formatting is used, rules should be documented and shared to improve the code detection mechanism

<u>Generating Codes in the DECE Common Redemption System (CRS):</u>

CRS supports generation of codes that adhere to the UltraViolet Common Code Format (CCF).

CRS stores bad words filters as a list of strings. CCF codes generated by the CRS are compared against this list. Codes that are a match are not pushed to the system, a new code is generated. Initial bad word filter only removes vowels from the CCF codes generated by the CRS: "A", "a", "E", "e", "I", "lower case I", "O", "o", "U", "u".

Studio requesting above and beyond this baseline CCF implementation can negotiate studio-specific requirements directly with Deluxe. Resulting code must still adhere to the requirements specified in this document.

<u>Detailed Checksum Calculation Instructions:</u>

UltraViolet Common Code Format supports both all-numeric and alphanumeric configurations. To handle non-numeric characters, all alphabetic characters are first converted to uppercase, then each character (numeric or alphabetic) is converted to an ASCII offset from '0' (ASCII 48) to get a numeric value for computation per the Luhn algorithm. (Note that this defines a codespace of 43, including unused alphanumeric characters and unused ASCII characters : through @, which is slightly inefficient but makes for simpler calculation.)

A check digit (0-9) is calculated, not a check character.

Below is an explanation and pseudo code on how to generate the check digit for a code:

- For the second-to-last (2nd from the right) character and every other (even-positioned from the end) character moving to the left, add ASCII value - 48 to the running total.
- Non-numeric characters add values > 10 to the running total. For example, 'M' is ASCII 77. Since 77 - 48 = 29, add 29 to the running total (not 2 + 9 = 11).
- For the rightmost character and every other (odd-positioned from the end) character moving to the left, use the formula 2 * n - 9 × INT(n/5) (where n = ASCII(character)-48 and INT() rounds off to the next lowest whole number) to calculate the contribution of every other character.
- Note on alphabetic characters: If you use the formula above on the numbers 0 to 9, you will see that it's the same as doubling the value and then adding the resulting digits together. For example, using 8, 2 × 8 = 16 and 1 + 6 = 7, or using the formula, 2 × 8 - 9 × INT(8/5) = 16 - 9 × 1 = 16 - 9 = 7 – identical to the Luhn algorithm. Using ASCII code points accommodates non-numeric characters as well by simply plugging ASCII value - 48 into the formula. For example, 'Z' is ASCII 90. 90 - 48 = 42 and 2 × 42 - 9 × INT(42/5) = 84 - 9 × 8 = 84 - 72 = 12. So we add 12 (not 1 + 2 = 3) to the running total.

Example for the identifier "139MT":

- **T** (ASCII 84) -> 84 - 48 = 36 -> 2 × 36 - 9 × INT(36/5) = 72 - 9 × 7 = 72 - 63 = **9**
- **M** (ASCII 77) -> 77 - 48 = **29**
- **9** (ASCII 57) -> 57 - 48 = 9 -> 2 × 9 - 9 x INT(9/5) = 18 - 9 × 1 = 18 - 9 = **9**
- **3** (ASCII 51) -> 51 - 48 = **3**
- **1** (ASCII 49) -> 49 - 48 = 1 -> 2 × 1 - 9 × INT(1/5) = 2 - 9 × 0 = **2**

Summing the results we get 9 + 29 + 9 + 3 + 2 = 52. The next number divisible by ten is 60. So our check digit (the difference) is **8**. (This can also be calculated as 10 - sum modulo 10, so 10 - 52 Mod 10 = 8.)

<u>Other considerations:</u>

1. The same unique ID can be generated by multiple studios, but the prefix will be different so there is no risk of code collision.
2. Any desire for code recycling should be discussed with DECE as it may have implications for CRS.