Coding/decoding processes and algorithms for identifiers made of alphanumeric segments lists, these segments being variable both in number and length, on an overall fixed size word.

Abstract

This invention is about codes, or identifiers, that can be described as a list of alphanumeric segments, such as for instance the GS1 identifiers, ISBNs for books, ISANs for audiovisual works, products or logistics codes, or the multiplicity of such codes used in different public or private organizations.

It describes algorithms and processes to code/decode (or write/store/read) them in a way that would alleviate current limitations regarding their structures, the idea being to provide in return, a greatly enhanced flexibility for their distribution and usage.

Typically these codes have today, for a given code space, a fixed number of segments, and in order to separate the different segments in a given code word, two techniques are used:

1. Each segment is allocated a given number of digits (or octets, or bits) in the definition of the code space, and the "content" for a given code segment instance is "pushed to the right" (or left) in the allocated defined length, with "bit stuffing" or leading zeros at the beginning of the segment.
2. Or rules, and associated tables, are used to determine, depending on the value of the first digits of a given code word, how the next segments have to be separated.

These two methods can be intermixed in a given code space. The first one is typically the one used for most GS1 identifiers (and the vast majority of other multisegment code spaces), as is for instance described in the "GS1 General Specifications v.7.0" (see for instance http://www.gs1uk.org/EANUCC/WORD_Files/GS_Section_1_V7.doc), the second one being used for instance for the ISBN code space as described in the ISBN's users' Manual (http://www.isbn-international.org/en/manual.html).

These two methods allowed to "deploy" these code spaces, codes that are essential in our ability to handle the corresponding objects or concepts, as well as in the ability to write the systems managing them: Library and bookshops would probably be quite impossible to run today without the ISBN code space for instance, or UNICODE is essential for multilingual scripts.

However the current structures of these codes impose quite serious constraints on their distribution and usage : The fixed number of segments usually also defines a distribution structure with a fixed number of levels, the fixed segment length imposing to "bet" on the proper length of a segment corresponding to its probable future use, and when using the second method it implies to maintain and share "deciphering" tables for the system to function, as is for instance expressed in the following ISBN users manual extract :

« Ranges are allocated according to anticipated demand (i.e., the size of the publishing programme) within a particular regional or language grouping. Some registration group elements are held in reserve by the International ISBN Agency to ensure future capacity. The International ISBN Agency will allocate these as necessary when ranges are low in any particular grouping. Additionally, registrant ranges within a registration group that has already been allocated to a specific regional or language grouping may remain undefined to satisfy future needs.
Comprehensive details of registration groups and registrant range metadata is available from the International ISBN Agency and enables the validation of the current allocation of defined ranges. Not all registration group and registrant combinations are valid. The formulaic information (using the comprehensive group and registrant range metadata) required to split the ISBN into its constituent parts follows below. See *Check digit* section for information regarding check digit calculation and validation.
The number of digits in each of the ISBN elements for registration group, registrant, and publication varies in length, although the number of digits contained in these three elements is nine in total. These nine digits, together with the three-digit prefix element and the check digit, make up the 13-digit ISBN. The number of digits in the registration group and registrant elements will vary according to the publishing output of the registration group or registrant in question. Registration groups for which large output of monographic publications is anticipated will receive group numbers of one or two digits. Publishers with an expected large output of publications will be assigned registrant numbers of two or three digits.
Note: The number of digits specified and assigned for registrant groups and registrants within prefix element 978 cannot be relied upon to predict those which are specified and assigned within future prefix elements (e.g., prefix element 979). Registration groups and registrant allocations for future prefixes will reflect assignment history and assignment projections for the entire prefix element system viewed collectively.
Determining the internal divisions of the 13-digit ISBN is a two-step process: first, determine the registration group using the rules for prefix elements assigned for ISBN; second, determine the registrant and publication element lengths using the registration group rules. Registration group rules are available from the International ISBN Agency. Table 1 illustrates the distribution of registration group ranges within prefix element 978. Any other EAN.UCC prefix defined for use within the ISBN system will have registration group rules available from the International ISBN Agency prior to any registration group assignment within that EAN.UCC prefix. *It is strongly recommended to check with the International ISBN Agency on a regular basis for possible additions or changes to registration group rules.*"

Also, "grouping" different code spaces always raises issues, due to the fixed number of segments.

Moreover, in the current digital context, there is a true "explosion" in the identification needs (identification needs which are exactly the same be it digital or not), and such "administrative sources of codes" could also be used for any kind of digital object, be it related to content or to the "computing infrastructure" itself. The "explosion" in identification needs is also present through the use of bar codes or RFIDs regarding a growing multiplicity of objects or messages. Some solutions are provided with the DOI for instance (see DOI.org) or through the use of UUIDs in various operating systems, but the use of the present invention, would most probably allow much more flexible distribution structures and ability to group existing code spaces, while maintaining the possibility to have fixed sized representation of these codes, which remain a key aspect in many treatments or storage means, as well as in the ability to use these codes for "deep identifiers" in a uniform fashion.

The principle is the following: Let's consider an identifier as a list of the form $n_1.n_2....n_p$, where if p is less than a given maximum M for a given code space. On this let's remark that M doesn't need to be "very big", indeed we are here talking about codes, whose structures are not meant to express classification or semantics, or to be s-expressions, "the flatter the better", the structure mirroring in fact the constraints of the distribution, something around 7 being most probably sufficient for all code spaces.

We consider that the "overall length" to write the identifier is L (this can be considered with bits, octets, or other). L can be seen as defining a format, and there also can be several of them.

We then define an algorithm to count and assign a number to each possible structure within L, and an inverse algorithm to deduct a structure from a given number. That is if L is 16 for instance, 1st segment 3 octets, 2nd segment 4, 3rd 9 , will be a structure and assigned a number by the algorithm, as well as 1st segment 3 octets, 2nd segment 4, 3rd 5, 4th segment 4.

Then, for a given instance of an identifier, first its "ideal" structure is computed (that is the length required for each segment without "leading zeroes" or "bit stuffing"), then the corresponding number for this structure is computed, and the identifier is then written as the concatenation of the structure number and the values of its segments.

Inversely, when reading an identifier, first the structure is deducted from the structure number, and then it is used to "retrieve" the identifier segments (and in many cases the identifier can of course be used without the need to separate its segments).

The end result is quite satisfactory as, for instance if we consider a code space of 16 octets and 7 segments maximum, only two octets are necessary to write the structure number, so that 14 octets are left for any identifiers, from 1 to 7 segments, without any restriction on segments size except for an overall "overflow". The objective being here in any case not "data compression", but greatly enhanced distribution flexibility while preserving an overall fixed written length.

Of course the use of this invention would not alleviate the need for central naming authorities responsible for the distribution of shared code spaces (managing the first segments) or the equivalent in private contexts, however we believe that its use could greatly enhance the overall dynamic and flexibility. With respect to its use for "computing objects", it could also allow to set up and write distributed infrastructures with a much higher degree of connectivity, as well as, perhaps even more importantly, provide the ability to change these infrastructures more easily.