

File naming convention using ONIX list codes

v0.3	29/3/2011	Incorporates minor edits for clarity, no changes to substance of proposal
v0.2	23/6/2010	Original proposal by Jesús Peraita

Introduction

This proposal deals with two potential problems that arise when organizations exchange bibliographic *resource files* – files such as product images or content samples.

Within the context of an ONIX for Books data feed, the ONIX metadata normally includes a URL from which a resource may be downloaded – but it may simply contain a filename, which needs to be matched with a file supplied separately. Alternatively, resource files may be exchanged without any accompanying metadata feed. The separate supply of the resource files raises two potential issues:

- *File name collision*

This might occur, for instance, when different publishers upload their resource files to the same repository. If two publishers use the same name for two different files, one will overwrite the other.

For instance, if two publishers use the name *quixote.jpg* for the cover image of two different books related to Don Quixote, a file name collision will occur and one of the files will be lost.

This problem might be alleviated at the repository level if separate directories are kept for each individual publisher or imprint. However, it is impossible to assure that this separation will exist throughout the complete book information chain. For instance, a distributor or a bookseller gathering data from a books-in-print repository might not be able to maintain this multiple directory structure and the file name collision will occur in his system.

- *Orphan resource files*

In general, resource files will be referenced in a higher-level bibliographic context, for instance in a <MediaFile> composite within an ONIX 2.1 <Product> record. This record provides additional information, e.g. about the kind of resource (via the <MediaFileTypeCode> element) and the product to which it relates (via the <ProductIdentifier> composite). In that way, a particular *quixote.jpg* file might be identified as a high-res front cover image of a book with a given ISBN.

However, this information is only available if the ONIX record is accessible and can be parsed. The resource file, by itself, is an *orphan file*. There is no way to find out what its content is or to what product it relates.

There are two additional issues:

- *Multiple versions of the same resource*

This is the case were two or more examples of the same resource *coexist*. For instance there might be two different photographs of the same author, or two different images of a front cover (maybe with different pixel sizes).

In this case, any party having access to these files must know that both of them are valid.

- *Time stamped resources*

This occurs when a resource file is *replaced* by a new example of the same resource: e.g. a JPEG cover image is replaced by a revised cover design. In this case, the user should be able to discover that the old version has to be replaced by the new one.

These are common problems faced at DILVE (www.dilve.es, the Spanish Books-in-print repository), where publishers upload resource files to a repository rather than providing an URL.

However, these problems can be encountered in any environment where resource files are stored or exchanged.

Scope

There are many circumstances where the abovementioned problems can surface, and a whole range of different solutions might be adopted. This paper does not propose a universal solution, but rather an easy-to-implement and realistic convention within a limited scope of application.

The proposal is based on the following premises:

- 1 In principle the file naming convention will only be applicable to files that can be referenced from within ONIX <Product> records, and associated with an ISBN. These resource files might be referenced in <MediaFile> or <OtherText> in ONIX 2.1, or in <SupportingResource> in ONIX 3.0.
- 2 Given its name, it should be easy to identify the 'type' of a resource, and relate a product to a resource file. It should also be trivial to parse the resource name to automate any processing required of the resource file.
- 3 The filename should not include information about characteristics of the file that can be discovered by analysing its content.

This is a compromise. For instance, it would seem that including the pixel size of an image in its filename would be useful. However, given that we will have to deal with resources of many different types and with an assortment of concrete characteristics, this would soon degenerate into unpractical conventions: e.g. codec, duration, and frame rate and size for videos, duration and sampling rate for audio, etc.

- 3 There is no perfect solution to this problem. However, an imperfect solution is preferable to no solution at all.

Convention requirements

The following information should be available from the file name:

- The product to which the file relates. This will be identified by its ISBN-13 (GTIN-13).
- The general category of the resource contained in the file: e.g. image, audio, text, etc. This will be identified by including the number of the ONIX Code list that relates to the corresponding kind of resource, e.g. in ONIX 2.1 that would be List 38 for image, audio, and video files and List 33 for descriptions and other supporting text.
- The specific kind of resource contained in the file: e.g. the cover image or the table of contents. This will be identified by using a specific value from the corresponding ONIX Code list.
- An optional version number, allowing two or more files of the same kind to coexist
- An optional timestamp
- File format will be indicated by its extension. This, of course, does not allow for a very precise handling of different versions of one given format (e.g. different versions of PDF), but it is a rough indication that will be sufficient in most cases.

Additionally, the name should comply with the following requirements:

- Only digits (0–9), letters (a–z, A–Z) and the underscore character should be used in the file name (a subset of the ASCII character set that is universally acceptable in file names). It is well-known that Windows and Linux behave differently when dealing with mixed-case filenames so it is recommended to use either upper or lower case throughout, even for file extensions, and to avoid names in mixed case. In any case, recipients should treat files and filenames as if they are case-insensitive.
- The filename must be easily parsed into different particles (some of which might be optional), with well-defined meanings.
- The file extension will be preceded by a period. No other periods, and no other punctuation characters should appear in the file name.
- Although there might be some legacy systems with severe filename length limitations the tendency is to move towards unlimited lengths. Therefore, this proposal will ignore such possible limitations.

File naming convention

The proposed resource file naming convention adheres to the following format:

`nnnnnnnnnnnnnnn_Looo_ccc_Vrr_Dyyyymmdd.ext`

where:

nnnnnnnnnnnnnnn ISBN13 (GTIN-13) of the corresponding product. No hyphens or other separator should be included.

L The letter "L". This may be lower or upper case.

_ A literal underscore character. **ooo** Number of an applicable ONIX Code list. Only digits 0–9 should be used. It must be immediately preceded by a letter "L". (ooo does not imply there should necessarily be three digits – eg List 38 may be indicated by “_L38_”)

ccc Code value from the ONIX List number "ooo". Characters a–z, A–Z and 0–9 are acceptable. (again, ccc doesn't imply that three characters are necessary – code 01 would be indicated by “_01_”. If the ONIX codelist specifies upper case letters, upper case should be used (and vice versa).

v The letter "V". This may be upper or lower case.

rr This and the preceding underscore and letter V specify an optional 'version' number, that should only be included in the case where two or more resources exist and their files share the same name. Only digits 0–9 should be used. (and for ease of sorting, always use two digits, with a leading zero if necessary). Note that 'versions' are intended to *co-exist*, so two different author photos or two different review texts may be supplied. Version 02 does not replace version 01. Use the D particle below to indicate replacement, using the same version number and a different date.)

The particle _Vnn is optional (but if omitted, treat as V01)

D The letter "D". This may be upper or lower case.

yyyymmdd A validity date for the resource file. It can be assumed that if two resource files names differ only in their validity dates, the newest one replaces any previous instance. If a resource file with a validity date in the past is uploaded, it should be used immediately, and should replace any older variation of the same resource. If a resource file with a validity date in the future is supplied, it should not be used immediately, but should be brought into use on the indicated date (and any older variation of the same resource should be retained until that date)

The particle _Dyyyymmdd is optional (but if omitted, treat as valid immediately)

.ext File extension. This will give a *hint* about the format of the file.

Examples

`9788496479357_L33_07.html`

This file contains a review text (code 07 in List 33) of the book with ISBN 9788496479357, in HTML format. Note that the file name, by itself, does not indicate if the file is XHTML compliant.

`9788496479357_L33_24.xml`

First chapter (code 24 in List 33) of the same book, in XML.

`9788496479357_L38_04.jpg`

Front cover image (code 04 in List 38) of the same book, in JPEG format.

9788496479357_L38_04_D20091231.jpg
9788496479357_L38_04_D20100623.jpg

Two versions of the front cover image (code 04 in List 38) of the same book, in JPEG format. The second one, dated 23/06/2010, supersedes the first one, dated 31/12/2009.

9788496479357_L38_04_v01.jpg
9788496479357_L38_04_v02.jpg

Two versions of the front cover image (code 04 in List 38) of the same book, in JPEG format.

9788496479357_L38_04.jpg
9788496479357_L38_04_v02.jpg

A variation of the last example, where V01 must be assumed for the first file.

9788496479357_L38_04.jpg
9788496479357_L38_04_v01.jpg

Yet one more variation of the last example, where V01 must be assumed for the first file, and thus the second file must replace the first immediately (since the D particle is missing)

Jesús Peraita
DILVE & Onix National Group, Spain