



A Practical Guide to Implementing Onix

| | |
|--|----|
| Introduction | 4 |
| So Where to Start? | 5 |
| So What Really is an Onix Message? | 6 |
| Rules of XML | 10 |
| XML Rule 1 | 10 |
| XML Rule 2 | 10 |
| XML Rule 3 | 10 |
| DTDs | 11 |
| Overview Document | 13 |
| XML Message Document | 13 |
| Product | 13 |
| Main & Sub-series | 14 |
| The Basic XML Structure of an ONIX Message | 18 |
| Five Steps to Onix | 21 |
| Step 1 | 21 |
| Step 2 | 23 |
| Step 3 | 24 |
| Step 4 | 24 |

| | |
|--------------------|----|
| Step 5 | 25 |
| Things to Remember | 27 |
| A Sample Message | 28 |

Introduction

Who would have thought that a four letter acronym like ONIX could cause so much confusion and consternation? ONIX stands for **ON**line **In**-formation **eX**change and it is “the international standard for representing and communicating book industry product information in electronic form” according to EDItEUR, who are the keepers of the standard (see www.editeur.org).

OK, so there is a standard for passing bibliographic information between interested parties and if you have started to look into this topic you will have found numerous sites that espouse the benefits of the standard and either stop there or launch into a swathe of technical details that would appear to challenge even the most gifted MENSA student. Either people take a top down view talking about benefits for supply chains and give vague cost benefits or they drop straight into technical details talking about required values for XML tag <j123>, etc. What is missing is a simple guide that talks a publisher through the details of the ONIX standard and gives them practical guidance on how to implement it. To this end we humbly give you the Practical Guide to Implementing ONIX.

This guide will, indeed has to, give a slightly technical (don't worry we'll be gentle) overview of the ONIX standard and the technologies it's based on. We will then talk you through five basic steps you will need to follow to get your title information out into the world in ONIX form.

Once you have an understanding of what is involved you will hopefully be better prepared to look into the market for a software solution that suits your businesses requirements or for the braver, more technically minded, souls give a basis for creating your own in-house solutions.

Now we will be discussing some quite technical issues and using I.T. terminology that may not be familiar to you but we will try and keep the acronyms down to a minimum and describe some of the more technical stuff in layman's terms.

What that does mean though is that some of you more technically minded people out there may take issue with some of the definitions we use to describe some of the ideas and technologies involved. We make no apologies about this. The aim of this document is to help non-technical publishers get a handle on ONIX and we may occasionally over-simplify some of the concepts and use terminology that may not be totally accurate but we take the view that simple is best and academic rigor will not get us where we need to be.

So Where to Start?

ONIX is a technical standard and as such requires a technical solution but don't be fooled, as you will see shortly, when you distill it down we are actually talking about just sending text files between interested parties. Fair enough, the text files do have to be formatted in a very specific way, and there are many ways to implement technologies to achieve the desired results, but they are just text files! The complexities arise because you have to split down and categorize your title information in very specific ways so that it can be bundled up into an ONIX message (which is just the text file) and submitted.

Unfortunately, it is this breaking down and linking of title information that may lead you to reviewing your whole business process and possibly even purchasing some title management software to pull your title information together! Don't panic just yet but please bear in mind that ONIX compliance is not a trivial task.

So our first job is to look at the ONIX standard and show you how to interpret the technical standards documentation that is available. We will then describe an ONIX message in terms of how it is structured. Once you understand what ONIX is and how it works the next step is to actually implement an ONIX solution for your business. To this end we have put together a framework of five steps that we believe you should follow to allow your business to successfully start sending ONIX messages.

Throughout the document we've throw in hints, tips and ideas, based on our own experiences, and where appropriate pointed you to other web sites and documentation to help try and make the whole process as painless as possible.

So What Really is an Onix Message?

Put simply an ONIX message is a text file that is formatted in a very specific agreed way so that anyone who receives the file will know exactly how to interpret the contents. More formally, the ONIX message is a set of data elements defined by 'tags' written in XML.

Whoa, backup. XML? Tags?

XML (eXtensible Markup Language) is a text based markup language for structuring information. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays. For example, content in a section heading has a different meaning from content in a footnote, which means something different than content in a figure caption or content in a database table, etc. Almost all documents have some structure and XML is a standard that allows the document/data structure to be absolutely defined.

If you're not 100% clear about what XML is or what it does then peruse a few of the definitions that the Google search engine brings up <http://www.google.com/search?num=100&q=define:XML>.

In XML, "tags" or "elements" (people tend to use these terms interchangeably) are used to delimit the beginning and end of a piece of data and these tags can be grouped and structured to meaningfully represent data.

For example the following address:

22 Arcacia Avenue
Little Hampton
Dorset
D45 9JK

could be represented in XML as follows:

```
<Address>  
  <AddressLine1>22 Arcacia Avenue</AddressLine1>  
  <AddressLine2>Little Hampton</AddressLine3>
```

```
<AddressLine3>Dorset</AddressLine3>
```

```
<PostCode>D45 9JK</PostCode>
```

```
</Address>
```

The tags are the named elements in the angled brackets e.g. `<Address></Address>` with the closing tag always starting with a `</`. Anything between the tags can then either be content, as in the text "22 Arcacia Avenue" between the `<Address1></Address1>` tags, or another tag.

You will note that the `AddressLine1`, `AddressLine2`, `AddressLine3` and `PostCode` tags are all encapsulated within the `AddressLine` tag fields in the sample above. By putting tags within tags complex data structures can be built up. The structure that has been built up with the XML tags for the address example is commonly called a "composite".

In addition to storing content between elements it is also possible to store information within the tag itself in the form of "attributes". For instance, in the example below a `textformat` attribute has been added and has been set to a value of `01`. Attributes are useful for adding ancillary information about tag content.

```
<Address1 textformat=01>22 Arcacia Avenue</Address1>
```

Attributes are not used heavily in the ONIX messaging standard but they do appear.

If you have never seen an ONIX message before refer to the 'A Sample Onix Message' section where you can see how XML is used to structure title information using tags and composites.

In and of itself the ability for XML to structure and describe content in this way is useful but the real power lies in the fact that the structure of an XML document is defined by something known as a Document Type Definition (DTD). We don't need to know the technical details of how a DTD works only that it is itself a text document that specifies rules on how an XML document should be structured and in particular how data elements are ordered and are interrelated. An XML document can be explicitly checked or "parsed" against a DTD and if its structure is somehow different from what the DTD expects an error message is produced and the XML document is considered "invalid".

If two parties have access to a DTD they know all the rules for an XML document (that conforms to that DTD) and so one party can pass information to the other knowing that if they use the same DTD they will unambiguously be able to interpret the XML content that is passed e.g. There will always be three address lines followed by a postcode, etc.

And that is exactly what happens with ONIX messaging, a DTD is maintained by EDItEUR, that is made available to all publishers, distributors, trade partners and interested parties who use it to unambiguously pass bibliographic information between themselves. The DTD effectively enshrines the ONIX standard in as far as it dictates what elements must be included and in what order.

By breaking down all of the information about a title into unambiguous tags of data and bundling them into an ONIX message information flow between computer systems can be more easily automated which theoretically makes the process cheaper and more efficient.

For example:

Consider a simple book title
"A history of Dungarvan - town and district"

The above can actually be split down so that computer systems can better sort and format it:

Title prefix = "A"
Title without prefix = "history of Dungarvan"
Subtitle = "town and history"
Title text case = "Sentence Case"

In an ONIX message the book title information looks like this:

```
<Title>
  <TitleType>01</TitleType>
  <TitleTextCase>01</TitleTextCase>
  <TitleText>A history of Dungarvan</TitleText>
  <TitlePrefix>A</TitlePrefix>
  <TitleWithoutPrefix>history of Dungarvan</TitleWithoutPrefix>
  <Subtitle>town and district</Subtitle>
</Title>
```

Since the title name is now split into constituent components the recipient can more easily manipulate the information. Maybe they could use the TitleWithoutPrefix tag content to sort titles more sensibly, whereas the TitleText element may be more appropriate for display on a web site or in an advertisement. It sounds obvious but before ONIX came along everybody was passing information around as big lumps of text, in different formats, so you either displayed a title as it came (if you could find it in the file), you re-keyed the information or you displayed nothing at all.

The down side to splitting out title information is that there is lots of it. We generated 8 lines of XML just to store a title name. Now imagine doing the same thing for all of your contributor, supplier, pricing, rights and categorization information for each of your titles and you'll start to appreciate the scale of the problem. There can be literally hundreds of XML tags to describe one title! Indeed the sample ONIX message in the 'A Sample Onix Message' section is for only one title and could be considered moderately rich in terms of the information being sent and

probably covers about 60% of the possible ONIX tags available.

By now you are probably realizing that it is simply not practical to do ONIX messaging without some software that can properly structure and centralise your title information and ideally automatically generate XML.

Rules of XML

As we've already mentioned an ONIX message is an XML document that conforms to a DTD maintained by EDItEUR. However, XML has a few basic rules of its own that must be adhered to if an XML document is to be considered correct or "well formed" and if you can get your head around them you'll find ONIX messages fairly straightforward to understand (though not necessarily straightforward to implement as we'll find out later).

XML Rule 1

Every opening tag or element must have an equivalent closing tag.

For example the following would be invalid as there is no closing Address1 tag

```
<Address>  
    <Address1>22 Arcacia Avenue  
</Address>
```

XML Rule 2

Element names are case sensitive so <Tag1> and <tag1> are different. Note the content text between the tags can be whatever case you desire.

XML Rule 3

Certain characters are not allowed in XML content and have to be replaced or "escaped" if the XML is to be considered well formed. In particular the '&' and '<' characters must be converted if they appear between tags. The '&' character should be replaced with the special code (or "escape sequence") & and the '<' should be replaced with the escape sequence <

Once an intended recipient loads the XML file and extracts the content between the tags they will convert the escape sequence characters back to their original character representations so that the text is displayed correctly.

In the following sample XML,

```
<tag1>Barnes & Noble</tag1> ,
```

the & symbol should be replaced by & to look like this

© 2005 Anko Publishing Software Limited

`<tag1>Barnes & Noble</tag1>`.

Once the XML had been transmitted, the recipient would then extract the 'Barnes & Noble' text and replace the & with the & symbol so that it looked like this 'Barnes & Noble' again.

Note the < characters that make up the tag names do not have to be escaped, only the content between the tags.

Valid characters that can appear in an XML document include:

- Space character
- Capital letters: A – Z
- Lower-case letters: a – z
- Digits: 0 – 9
- Punctuation and brackets: !"(),-.:;?[]{}
- Currency, arithmetic, computer and other symbols: # \$ % * + / = > \ @ _ ` | ~

Any characters not included above such as foreign characters or scientific symbols may need to be "escaped" with appropriate escape sequences.

DTDs

So you know the basic rules for an XML document the only other thing you need to know about an XML document is how you link it to a DTD (which as we now know has all the rules that dictate how the XML elements should be structured).

The link is achieved by adding a special DOCTYPE tag at the top of the XML document

```
<!DOCTYPE ONIXMessage SYSTEM "http://www.editeur.org/onix/2.1/02/reference/onix-international.dtd">
```

The significant bit is "http://www.editeur.org/onix/2.1/02/reference/onix-international.dtd" as this is the path to the DTD, which in this case is stored on the EDItEUR web server.

The number 2.1 in the text actually refers to the version of ONIX to be used whilst the following 02 refers to the sub-version.

Sub-versions represent minor modifications that have occurred in the ONIX standard, between major releases (see the DTD declaration section of the XML Message documentation for a discussion about versions), such as tags names changing or minor structural changes to the XML structure.

You must bear in mind that the ONIX standard is constantly evolving and people from all over the world are contributing ideas and making suggestions that help fix problems they come across or improve the standard as they try and implement ONIX within their own organisations. A Yahoo forum has been setup by EDItEUR http://groups.yahoo.com/group/ONIX_IMPLEMENT/ which in their words:

"will act as the forum for asking questions about the interpretation of ONIX standards, raising any practical problems which need to be addressed in future releases, and participating in discussion aimed at finding the best solutions"

Any ideas or issues that come through the forum are discussed by national ONIX committees and if approved are included in the next release of the standard.

We would strongly recommend you sign up to the forum as it has a complete history of issues and discussions about the standard and the chances are that any technical questions you may have will have already been answered within the forum. You can also see the kinds of issues being discussed today which may become parts of the standard tomorrow, although EDItEUR have stated that they do not foresee any major revisions of the standard in the near future (more tweaking around the edges)

When a release comes out a new version of the ONIX DTD is created (with a new sub-version number) to ensure backward compatibility since not everybody will have made the necessary changes at the same time. EDItEUR keep all versions of the DTDs available on their web site.

One of the most contentious areas about the ONIX standard is this wide variation of versions and interpretations. As the standard has evolved the structure of ONIX messages has changed and at any given time different people are able to produce and receive different versions of an ONIX message. So unfortunately, instead of being able to produce a single ONIX message you can send to everyone you have to produce different versions for different recipients and the "standard" isn't quite as standard as one might have hoped. Practically this means that you can't simply create a one off project to "do ONIX" but rather must put flexible systems in place that will allow you to modify our ONIX message feeds in synch with the standards and your recipients requirements for data as they change over time.

Onix Documentation

The bulk of the EDItEUR technical information is held in PDF documents, which list all of the ONIX tags available along with short descriptions on their use (see <http://www.editeur.org/onix.html>). The documents are fairly technical in nature and are written primarily for development staff so you may find them a little intimidating, but hey that's why we wrote this!

There is a set of documentation for each version of ONIX. For the latest version 2.1, there are actually five PDF documents covering:

- Overview
- XML Message
- Product
- Main Series
- Sub-series

Overview Document

The Overview document is exactly that and should be your first port of call. Hopefully by the time you have read this guide you won't find anything too controversial in there and it will simply be reinforcing what you already know.

XML Message Document

The XML message document discusses the overall structure of an ONIX message and describes some of the technical XML requirements. This document also defines the elements required for the ONIX message header (see later).

Product

The Product document is the biggest of the documents (180 pages) and contains the majority of ONIX element definitions that you will be using to generate your ONIX messages. Don't panic though as not all of the ONIX fields are necessarily required, in fact you could theoretically get away with only sending half a dozen fields for each title. The standard is designed to cover a lot (we hesitate to use the word 'all' as the standard is constantly being updated) of supply chain eventualities. You should check with your intended trade partner/distributor and get a list of fields they require and take from the documentation what you need.

The document is split down into 26 sub-sections using a PR.1, PR.2, ...PR.26 reference system which describe different groupings of tags, for instance contributors, series, pricing, etc. and each sub-section has its own numbering e.g. PR.1.1, PR.1.2, PR.1.3, etc. that map to the individual tags for that section.

The example below is from the PR.22 section of the Product documentation that covers Dimensions of a title, specifically a measure unit code.

PR.22.3 Measure unit code

An ONIX code indicating the measure unit in which dimensions are given. Mandatory in each occurrence of the <Measure> composite, and non-repeating. This element must follow the dimension to which the measure unit applies. See example below.

| | |
|----------------|---------------------------|
| Format | Fixed-length, two letters |
| Code list | List 50 |
| Reference name | <MeasureUnitCode> |
| Short tag | <c095> |
| Example | <i>mm</i> |

This "PR" reference system is sometimes quoted by distributors when communicating their ONIX message requirements so you may see text like "*PR.1.1 is mandatory*", where they are stating that the first tag in the PR.1 section of the Product documentation (which happens to be 'Record Reference Number' tag in this case) is mandatory. You will find yourself referring to this document repeatedly when we go on to discuss the ONIX message structure in more detail.

Main & Sub-series

The Main Series and sub-series documents cover a very specific "*application requirement for the German ONIX user group*", and so is unlikely to be of interest to you unless of course you are dealing with such specific German groups. If this is the case, you will find that the documents follow a similar format to the main product document and simply describe yet more tags for your ONIX messages.

In addition to the PDF files you will have downloaded a series of html files e.g. onix-codelist-1.htm, that contain details of ONIX codes that should be used for certain ONIX tags. The Code Lists are referenced from within the PDF documents as required (see below).

To see the list of possible values for this tag you would open the onix-codelist-3.htm code list file.

PR.1.5 Record source type code

An ONIX code which indicates the type of source which has issued the ONIX record. Optional and non-repeating, independently of the occurrence of any other field.

| | |
|----------------|----------------------------------|
| Format | Fixed-length, two numeric digits |
| Code list | List 3 |
| Reference name | <RecordSourceType> |
| Short tag | <a194> |
| Example | 01 |

If you then open the equivalent html code list file, i.e. in the above example List 3 equates to onix-codelist-3.htm, in your web browser you will see something similar to the screenshot below.

ONIX Code Lists Issue 2, December 2003

List 3: Record source type code

| Value | Description | Notes |
|-------|-------------------------|---|
| 00 | Unspecified | |
| 01 | Publisher | |
| 02 | Publisher's distributor | Use only for a distributor appointed by the publisher, as distinct from a wholesaler. |
| 03 | Wholesaler | |
| 04 | Bibliographic agency | |
| 05 | Library bookseller | |

The value column on the left represents the actual code that should be sent in the specified ONIX message tag.

If you look closely at any of the tag definitions within the EDItEUR ONIX standard documents you will see that there are in fact two definitions for each tag (see screen shot below).

PR.13.1 BISAC main subject category

A BISAC subject category code which identifies the main subject of the product. Optional and non-repeating. Additional BISAC subject category codes may be sent using the **<Subject>** composite. *Note that the data element reference name was assigned during a period when the BISAC name had been changed to "BASIC". The name has now officially reverted to "BISAC", but the ONIX data element name cannot be changed for reasons of upwards compatibility.*

| | |
|----------------|--|
| Format | Fixed-length, three upper-case letters and six numeric digits. |
| Code list | BISAC Subject Heading Codes Please contact info@bisg.org for details, or check the BISG website at http://www.bisg.org/publications.html |
| Reference name | <BASICMainSubject> |
| Short tag | <b064> |
| Example | ARC007000 |

There are two definitions for each tag

A textual name referred to as the 'reference name' or 'long tags' along with a 'short tag' definition, which consists of a single character followed by a three-digit number (see examples of long and short tags in use below).

Long reference version tag sample:

<Header>

<FromCompany>Portadas.net</FromCompany>

<FromPerson>Bernie Rabow bernie.rabow@portadas.net </FromPerson> <ToCompany>EDItEUR</ToCompany>

<ToPerson>David Martin</ToPerson>

```
<MessageNumber>1213</MessageNumber>
```

```
<SentDate>200007311330</SentDate>
```

```
</Header>
```

Equivalent short version tags:

```
<header>
```

```
<m174>Portadas.net</m174>
```

```
<m175>Bernie Rabow bernie.rabow@portadas.net </m175>
```

```
<m178>EDItEUR</m178>
```

```
<m179>David Martin</m179>
```

```
<m180>1213</m180>
```

```
<m182>200007311330</m182>
```

```
</header>
```

Ideally everybody would use the reference name version of the tags as they are much more readable. Unfortunately, the majority of distributors and trading partners prefer the short tag version because the generated ONIX message files are smaller and can be processed more efficiently by their computer systems. This just means that if there is a problem with an ONIX message the feedback you get will quote short tag names, which you will have to lookup within the standards documentation (just another minor complication making your jobs harder or at least more technical).

OK so we know about the ONIX standard and we have all of the documentation so lets look at how ONIX messages are actually structured.

The Basic XML Structure of an ONIX Message

At the top of the document are two mandatory lines.

```
<?XML version="1.0" encoding="utf-8"?>
```

```
<!DOCTYPE ONIXMessage SYSTEM "http://www.editeur.org/.../onix-international.dtd">
```

The first is an XML specific tag. There is no need to worry why you just need to make sure its there – unless of course you are now into XML, in which case you can check the web for information about XML encoding.

The second line is the DTD declaration, which we've already discussed.

For those of you still with us, you might conclude that since we now know that there are in fact long and short versions of tags there would have to be two versions of the DTD, to validate the respective ONIX message tag formats. You would be correct. The example DTDs below are subtly different but if you were to attempt to validate a short tag XML message against the long version of the DTD, or vice versa, you would receive numerous errors about invalid or unrecognized tags.

```
<!DOCTYPE ONIXMessage SYSTEM "http://www.editeur.org/onix/2.1/reference/onix-international.dtd">
```

```
<!DOCTYPE ONIXmessage SYSTEM "http://www.editeur.org/onix/2.1/short/onix-international.dtd">
```

The natural extension to this need to reference the right DTD is that you may not be able to create just one ONIX message and send it to all of your trade partners/recipients. If one party requires long tags and somebody else requires a short tag version you will have to create two versions of your message (in fact if some recipients use older versions of the standard you may have to create even more versions of your message).

The next element in the ONIX Message is the first or "root" element of the XML document <ONIXMessage>. If you recall, we said that every XML tag must have an equivalent closing tag and so it is that this root element effectively encloses all of the other ONIX tags and has its closing tag at the very end of the document (see XML document structure outline below and sample ONIX message in the section 'A Sample Onix Message').

```
<ONIXMessage>  
  <Header> ————— Message Header Data Element  
  </Header>  
  <Product> ————— Product Information Data Elements for product 1  
  </Product>  
  <Product> ————— Product Information Data Elements for product 2  
  </Product>  
</ONIXMessage>
```

Next, every ONIX message must have a single header composite (i.e. a grouping of data elements) which contains details about who transmitted the ONIX message, the date the message was generated and additional, but optional, default parameters which can be used throughout the remainder of the document. See the EDItEUR XML Message documentation for details.

After the header we have one or more product tags, each of which contains details about each of your titles i.e. a product tag for each title.

Within the product tags you will find all of the tags that make up the title information as per the ONIX Product standard document.

Interestingly, a lot of distributors insist on any ONIX messages being formatted with only one tag per line within the file i.e. a carriage-return after each closing tag (like below)

```
<Title>  
<TitleType>01</TitleType>  
<TitleTextCase>01</TitleTextCase>  
<TitleText>A history of Dungarvan</TitleText>  
<TitlePrefix>A</TitlePrefix>  
<TitleWithoutPrefix>history of Dungarvan</TitleWithoutPrefix>
```

<Subtitle>town and district</Subtitle>

</Title>

as opposed to

<Title><TitleType>01</TitleType><TitleTextCase>01</TitleTextCase><TitleText>A history of
Dungarvan</TitleText><TitlePrefix>A</TitlePrefix><TitleWithoutPrefix>history of Dungarvan</TitleWithoutPrefix><Subtitle>town and
district</Subtitle></Title>

This is not a requirement of either the ONIX or XML standard and so must presumably be down to limitations in the computer systems the distributors are using? As we have already stated you need to check with the intended recipients and find out what they can/can't/won't accept in terms of ONIX as it will affect the systems you use to generate the messages.

Five Steps to Onix

We believe that there are basically five steps that you will have to go through to set up your systems/business so that you can generate ONIX messages.

The Basic Steps are:

- Match your data to the ONIX standard.
- Clean your data.
- Convert your data.
- Validate your ONIX messages.
- Submit your messages to your trade partners and distributors.

Step 1

As you will have seen by now full ONIX is a very detailed standard but it is a standard that is trying to cover a lot of bases and not all parts of the standard will be relevant to your business.

So the first step is to work out which elements of the standard you need to fully describe your titles and meet the requirements of your target ONIX message recipients.

As we've already said an ONIX message breaks down basic title information into discrete XML tags but it also includes additional status information such as product availability status and product form codes. These additional pieces of information will be something that you will have to explicitly add as they are ONIX specific codes and you would be unlikely to have equivalent status codes within your organisation. If you do have such codes you would have to check that they are correctly mapped to the ONIX equivalents.

Check out Booknet Canada and the Australian Publisher Association web sites as these guys are ahead of the game and have created help guides, notes and spreadsheets that will help you map your data with the appropriate ONIX tags.

Their approach of filling in the blanks on a spreadsheet will certainly crystallize your view of your title information. Suddenly, having price information held in one spreadsheet and blurbs held in word documents with rights information held in an old access database will become a real issue as you need to bring together all of the pieces of information together into a coherent form.

One area to watch out for when mapping your data is dependencies between ONIX tags. For instance, if you set the Product Form tag (PR.3.1) to 'DG', which is an electronic book text, you must include additional tags specified in the section PR.4 of the Product specification to supply more detail about the format of the electronic book. If you submit an ONIX message without the additional information the chances are that the recipient will reject the message (or have a grumble at least).

There are lots of other dependencies between tags that are kind of documented but they can sometimes be quite subtle and may only be picked up when you actually submit your first ONIX message. The good news is that once you've been through the cycle of submitting a message, having data errors reported, fixing data errors and re-submitting the message a few times you should not have problems with the same titles again. You'll find that apart from the occasional price change or marketing blurb update the bulk of the title information changes fairly infrequently and the data errors reported by your message recipients will drop off.

If you haven't got a single title management system you will be considering one after you have completed this exercise for the first time because you certainly won't want to repeat the process every time you need to send an ONIX message!

So now ONIX compliance suddenly is becoming a big deal because it can potentially become a major outlay requiring you to implement a large software product that affects your entire business.

More and more title management solutions and publishing software products are adding ONIX compliance to their list of features but there is usually a hefty price tag attached and ONIX is simply an output feature of a much bigger more complex system. So you will have step back and assess what you actually need and can afford.

In the longer term ONIX compliance will not be an option (trading partners may well simply refuse to accept anything else) but for the moment the cost benefits for implementing may not be completely obvious. Depending upon your distributors, and your relationship with them, you may currently find that there is a lag of a few weeks between you sending them title information (in whatever form) and it being made available to the outside world. ONIX messaging would reduce that gap down to days and there may be a competitive advantage to be had but once everyone is doing ONIX that advantage will disappear. Your argument might be that consistent data is the benefit but that will be dependent upon the software systems you use to generate the ONIX messages. Rubbish in equals rubbish out!

We're certainly not suggesting that you do not use ONIX, but in the short term certainly, you will be looking at spending a lot of time and money to step up to the plate for very little immediate benefit. The fact that you may end up with a centralised title management database

will reap massive benefits in the long term, but it's a fair bet you weren't considering buying software that will touch every part of your business before you started reading this!

Conversely, in the short term distributors may accept formatted Excel spreadsheets with ONIX data and you may be tempted to take this cheap option. However, this latitude won't last forever and you will be 'encouraged' to start generating correctly formatted XML ONIX messages. Besides if you attempt to just use a spreadsheet to submit ONIX information you'll have to either:

- manually keep your old systems in synch with the spreadsheet (for more than couple of dozen titles this can quickly become unworkable),
- use the spreadsheet as a central source of your title management (Excel doesn't share very well – first person to load a file can edit it, everyone else only gets a read-only copy) or,
- change your old systems to automatically generate the spreadsheets (but if you were going to do that you may as well generate XML messages anyway!).

So suddenly the simple need to create a formatted text file has caused you to review your entire business and what software it uses.

Assuming, you have got this far and you still intend on implementing ONIX you then need to move to step 2.

Step 2

As you work through your data you will need to fix or discard inconsistent, incorrect or incomplete data because the ONIX standard does not stand for ambiguity. If your data is not in the correct form the recipients will reject your ONIX messages, as they will almost certainly be using computers to automatically validate the information and as you know computers can be quite dumb as they can't "sort of" interpret what you mean. Its either on or off, yes or no, black or white.

So whilst you are looking at your data you will need to consider solving the main causes of "dirty" or incorrect data such as:

- Poor data entry
- Data missing from database fields
- Lack of company- or industry-wide coding standards
- Multiple databases in different departments & companies
- Older systems with poor or obsolete data

And lets make no bones about it this is the biggest most painful part! You will have done quite a bit of groundwork in step 1 but if you don't have a state of the art title management database you'll be wishing you did at the end of this process.

You won't find much help out there because it's your data! However, it is generally a one off exercise that will benefit your business greatly, even if you never send a single ONIX message. The act of reviewing business processes that you have done for years, by rote, may well bring on a realization that some of the processes are inefficient or even redundant and that you can use it as an opportunity to streamline the business. Think of it as spring-cleaning exercise.

Conversely, if you don't get this bit right you'll continually waste hundreds of man-hours dealing with rejected ONIX messages that contain invalid data!

Step 3

Now that you have your data cleansed and all the ONIX field tags mapped you can generate ONIX Messages, right?

Well nearly. Despite the fact that ONIX is a universal standard certain distributors have limitations on what form of ONIX they will support. As we mentioned earlier there are actually different versions of ONIX so you may have to produce different versions of ONIX for different recipients.

So you need to find out what your distributors/trade partners will accept. Then, you can generate the appropriate ONIX message putting your data into the right ONIX tags in the right sequence and format so that the message validates against the appropriate DTD.

Something to consider is whether you wish to be ONIX accredited. In the UK, check out <http://www.bic.org.uk/bbinfo.html> for details of the BIC ONIX accreditation scheme. In and of itself being accredited does not appear to actually have any tangible benefit, but if accreditation is your thing moving towards the standard should ensure your ONIX data will be suitable for submission to Nielsen Book Data who have worked closely with BIC and expect ONIX to this standard.

In Canada you can checkout BookNet Canada http://www.booknetcanada.ca/booknet/biblio_cert.shtml

Whilst for Australia got to http://www.publishers.asn.au/index.cfm?doc_id=201

Step 4

Once you finally manage to generate a message you will need to ensure that what you have produced is valid.

Now there is a subtle distinction between a valid message parsing against an appropriate DTD and what a distributor/recipient considers valid. The DTD will ensure that certain tags are present and in the right place, but it knows nothing about the data in between the tags. If you send

a price in pounds sterling but set the country to be Finland, as far as the DTD is concerned the ONIX may be valid because you have the price currency and country fields it expects but your distributor will almost certainly flag an issue with the data.

Before you send an ONIX message, as a minimum, you need to ensure it parses against the DTD. However, to check for data completeness you will have to revert to using internal data checks or possibly some third party tool that will check your ONIX message. Ultimately the actual recipients of the ONIX messages will dictate if a message is acceptable and should really be your first port of call.

There are a few web sites that will validate an ONIX message for you but they tend to be very technical and the messages will tersely tell you that for example "tag b089 was expected". It will be up to you work out which tags are causing the problem and then fix them and this is where your mapping exercise in step 1 would come into play. The mappings will allow you to trace which tag is causing the problem and hopefully why so that you can fix the source data. We warn you now though that when you submit your seemingly well-formed message to some of these validation tools you will get a barrage of error messages back and it will look like your message is totally useless. A lot of red herrings or irrelevant error messages may appear (possibly because you aren't using certain tags that are not relevant to your business) and you will have to work out which are things you actually have to deal with. If you haven't already registered with the ONIX_IMPLEMENT yahoo forum, we mentioned earlier, you might want to register with it at this point, as the chances are the problems/questions you have will have already been asked and tackled here. Also check out the Anko forums <http://www.anko.ie/bulletin/> where you will find lots of helpful information. A lot of the advice is specific to our products but you should find that it is scarily applicable to your situation.

Some examples of ONIX validation tools:

Booknet Canada have an ONIX Inspector that can be found at http://www.booknetcanada.ca/booknet/biblio_onix.shtml#000114

The Australian Publishers Association have one too http://www.publishers.asn.au/index.cfm?doc_id=319

Step 5

At Last! Well nearly.

Generally you can email or FTP your ONIX message to the intended recipient.

They will send you feed back if there is a problem with the data and for the first couple of submissions you may well get a mountain of stuff back. Don't despair though as you'll discover that a problem for one title may be an issue for others so fixing the source data once will clear lots of problems.

You will initially need to send a full title update and then only send incremental changes i.e. only titles that have changed since the last time you sent a message. In reality this means that the Notification Type tag (PR.1.2) will need to be modified for each title.

The ability to track what has changed to send incremental updates sounds simple but it's a major pain and should be high on your list of things to check for if you are reviewing possible ONIX software solutions. Especially when you consider that you may be sending ONIX messages to different recipients and will have to keep track of who has had what.

The primary reason for incremental updates is that distributors do not want to clog up their computer systems needlessly processing thousands of titles every time they receive a message if nothing has changed. Now if there are only a few hundred titles some distributors may just accept full updates each time rather than face the prospect of receiving nothing at all (because a publisher can't get an incremental system to work) but check first.

And that is it! The major up front work is complete! There will always be niggles with data and maybe the odd title which requires new tags you have not used before but if your software is reasonably robust there will not be too many problems. Slowly the standard will move on and you will have to update your XML outputs to fall in line as your distributors/recipients adopt the latest versions. One saving grace here is that it can take a fair while for the bigger distributors to upgrade to the latest standard simply because the scale of change for them is that much bigger and it potentially affects so many publishers.

Things to Remember

ONIX compliance is not a simple project that can just be completed. Pulling together title information into a coherent relational database system can be very costly both in terms of time and money and the standard keeps moving so you will have to keep up to date.

- The standard is geared for the whole supply chain and a valid ONIX message may consist of as little as a dozen tags or literally hundreds for each title.
- Don't let the ONIX tail wag the publishing dog - not all ONIX tags will make sense in your business and you may find yourself making up data to fill tags.
- Be guided by what your distributors, trading partners and customers need.
- If you are looking at purchasing a title management solution to implement ONIX remember that the costs of the actual software are likely to only represent a small proportion of the total cost. Check out our white paper on things to consider when purchasing title management software <http://www.anko.ie/Pages/General/ThingsToConsider.htm>

A Sample Message

Below is a sample Onix message using Onix 2.1 version 02 with the longer more descriptive tags:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ONIXMessage SYSTEM "http://www.editeur.org/onix/2.1/02/reference/onix-international.dtd">
<ONIXMessage>
<Header>
<FromEANNumber>123456789</FromEANNumber>
<FromCompany>Anko Publishing</FromCompany>
<FromPerson>Sara Elgerot</FromPerson>
<FromEmail>sara.elgerot@anko.ie</FromEmail>
<SentDate>20050623</SentDate>
<DefaultLanguageOfText>eng</DefaultLanguageOfText>
<DefaultPriceTypeCode>02</DefaultPriceTypeCode>
<DefaultCurrencyCode>EUR</DefaultCurrencyCode>
<DefaultClassOfTrade>General</DefaultClassOfTrade>
</Header>
<Product>
<RecordReference>T0142</RecordReference>
```

```
<NotificationType>03</NotificationType>
<ProductIdentifier>
<ProductIDType>02</ProductIDType>
<IDValue>0953481417</IDValue>
</ProductIdentifier>
<ProductIdentifier>
<ProductIDType>15</ProductIDType>
<IDValue>9780953481415</IDValue>
</ProductIdentifier>
<ProductIdentifier>
<ProductIDType>03</ProductIDType>
<IDValue>9780953481415</IDValue>
</ProductIdentifier>
<ProductIdentifier>
<ProductIDType>04</ProductIDType>
<IDValue>012345678905</IDValue>
</ProductIdentifier>
<Barcode>01</Barcode>
```

```
<ProductForm>BC</ProductForm>
<ProductFormDetail>B106</ProductFormDetail>
<NumberOfPieces>1</NumberOfPieces>
<Series>
<Title>
<TitleType>01</TitleType>
<TitleText>Waterford History</TitleText>
</Title>
<NumberWithinSeries>2</NumberWithinSeries>
</Series>
<Title>
<TitleType>01</TitleType>
<TitleText>A History of Dungarvan</TitleText>
<TitlePrefix>A</TitlePrefix>
<TitleWithoutPrefix>History of Dungarvan</TitleWithoutPrefix>
<Subtitle>Town and District</Subtitle>
</Title>
<Contributor>
```

<SequenceNumber>1</SequenceNumber>

<ContributorRole>A01</ContributorRole>

<PersonName>Patrick C Power</PersonName>

<PersonNameInverted>Power, Patrick C</PersonNameInverted>

<NamesBeforeKey>Patrick C</NamesBeforeKey>

<KeyNames>Power</KeyNames>

<BiographicalNote textformat="00">Patrick trained as a teacher and spent most of his life teaching career in Ballyneale. In 1964 he graduated BA, in 1965 he graduated MA and in 1971 he was conferred with a PhD. These degrees were won in University College Galway.</BiographicalNote>

</Contributor>

<ContributorStatement>Patrick trained as a teacher and spent most of his life teaching career in Ballyneale. In 1964 he graduated BA, in 1965 he graduated MA and in 1971 he was conferred with a PhD. These degrees were won in University College Galway.</ContributorStatement>

<EditionTypeCode>UXP</EditionTypeCode>

<EditionNumber>1</EditionNumber>

<Language>

<LanguageRole>01</LanguageRole>

<LanguageCode>eng</LanguageCode>

</Language>

<Language>

```
<LanguageRole>02</LanguageRole>
<LanguageCode>gle</LanguageCode>
</Language>
<NumberOfPages>320</NumberOfPages>
<Illustrations>
<IllustrationType>00</IllustrationType>
<Number>48</Number>
</Illustrations>
<BICMainSubject>YHBC</BICMainSubject>
<BICVersion>1.2</BICVersion>
<Subject>
<SubjectSchemeIdentifier>12</SubjectSchemeIdentifier>
<SubjectSchemeVersion>1.2</SubjectSchemeVersion>
<SubjectHeadingText>Local History</SubjectHeadingText>
</Subject>
<Subject>
<SubjectSchemeIdentifier>23</SubjectSchemeIdentifier>
<SubjectSchemeVersion>2</SubjectSchemeVersion>
```

```
<SubjectCode>Local Irish History</SubjectCode>  
<SubjectHeadingText>Local History</SubjectHeadingText>  
</Subject>  
<AudienceCode>01</AudienceCode>  
<OtherText>  
<TextTypeCode>01</TextTypeCode>  
<Text textformat="00">home tourism history urban renewal industry Le Tour De France contact
```

Dungarvan from the airDungarvan, the administrative headquarters for County Waterford, is a strategically located town on the south-east coast of Ireland.

In the late 1800's excavations at a local cave uncovered Ireland's only cache of Mammoth elephant bones. Some 40,000 years ago, Dungarvan Town provided ideal living conditions for woolly mammoths, horses, and giant ground sloths.

Stone age settlements have been found in the area around the town, but Dungarvan's history really began in the third century, when a tribe called the Deise settled there. The original Irish name of the town, Dun Garbhan (Garbhan's Fort) takes its name from Saint Garbhan, who founded a church there in the seventh century.

The town grew into an urban centre in the shelter of a large Anglo-Norman fortification founded in 1185. In 1215 the town was granted a charter by King John. Recent excavations have revealed much about Dungarvan's early history. As a town, it really only came into existence in

the Norman period. A church at the top of the town was linked to the castle by a single street - still called Church Street. A wall on the land side of the town ran from the church to the castle.

For centuries after, Dungarvan was noted for its thriving hake fishery. The town walls were extended to cover the seaward side, and a large commercial pottery operated just outside these new walls. The spectacular turreted castle of King John's time was radically redesigned in the 1400's to accommodate cannons. They weren't protection enough, for in the late 1400's the castle was captured - and blown up.</Text>

</OtherText>

<OtherText>

<TextTypeCode>02</TextTypeCode>

<Text textformat="00">An extensive review of the history of Dungarvan.</Text>

</OtherText>

<OtherText>

<TextTypeCode>07</TextTypeCode>

<Text textformat="00">Quite simply the best book on the history of Dungarvan</Text>

<TextSourceTitle>Waterford News</TextSourceTitle>

</OtherText>

<ProductWebsite>

<ProductWebsiteDescription textformat="00">home tourism history urban renewal industry Le Tour De France contact

Dungarvan from the airDungarvan, the administrative headquarters for County Waterford, is a strategically located town on the south-east coast of Ireland.

In the late 1800's excavations at a local cave uncovered Ireland's only cache of Mammoth elephant bones. Some 40,000 years ago, Dungarvan Town provided ideal living conditions for woolly mammoths, horses, and giant ground sloths.

Stone age settlements have been found in the area around the town, but Dungarvan's history really began in the third century, when a tribe called the Deise settled there. The original Irish name of the town, Dun Garbhan (Garbhan's Fort) takes its name from Saint Garbhan, who founded a church there in the seventh century.

The town grew into an urban centre in the shelter of a large Anglo-Norman fortification founded in 1185. In 1215 the town was granted a charter by King John. Recent excavations have revealed much about Dungarvan's early history. As a town, it really only came into existence in the Norman period. A church at the top of the town was linked to the castle by a single street - still called Church Street. A wall on the land side of the town ran from the church to the castle.

For centuries after, Dungarvan was noted for its thriving hake fishery. The town walls were extended to cover the seaward side, and a large commercial pottery operated just outside these new walls. The spectacular turreted castle of King John's time was radically redesigned in the 1400's to accommodate cannons. They weren't protection enough, for in the late 1400's the castle was captured - and blown up.</ProductWebsiteDescription>

<ProductWebsiteLink><http://www.anko-publishing.com></ProductWebsiteLink>

</ProductWebsite>

<Prize>

```
<PrizeName>Best Irish History</PrizeName>
<PrizeYear>2001</PrizeYear>
<PrizeCountry>IE</PrizeCountry>
<PrizeCode>02</PrizeCode>
<PrizeJury>Irish Historical Society</PrizeJury>
</Prize>
<Imprint>
<NameCodeType>02</NameCodeType>
<NameCodeTypeName>Proprietary</NameCodeTypeName>
<NameCodeValue>Anko</NameCodeValue>
<ImprintName>Anko Publishing Software Limited</ImprintName>
</Imprint>
<Publisher>
<PublishingRole>01</PublishingRole>
<PublisherName>Anko Publishing Software Limited</PublisherName>
</Publisher>
<Publisher>
<PublishingRole>01</PublishingRole>
```

```
<NameCodeType>02</NameCodeType>
<NameCodeTypeName>Anko Internal Publishing Code</NameCodeTypeName>
<NameCodeValue>123456</NameCodeValue>
<PublisherName>Anko Publishing Software Limited</PublisherName>
</Publisher>
<CityOfPublication>Waterford</CityOfPublication>
<CountryOfPublication>IE</CountryOfPublication>
<PublishingStatus>04</PublishingStatus>
<AnnouncementDate>20001105</AnnouncementDate>
<PublicationDate>20001105</PublicationDate>
<SalesRights>
<SalesRightsType>01</SalesRightsType>
<RightsCountry>IE</RightsCountry>
<RightsTerritory>ROW</RightsTerritory>
</SalesRights>
<SalesRights>
<SalesRightsType>02</SalesRightsType>
<RightsCountry>GB</RightsCountry>
```

```
</SalesRights>  
<SalesRights>  
<SalesRightsType>03</SalesRightsType>  
<RightsCountry>ZW</RightsCountry>  
</SalesRights>  
<Measure>  
<MeasureTypeCode>01</MeasureTypeCode>  
<Measurement>29</Measurement>  
<MeasureUnitCode>mm</MeasureUnitCode>  
</Measure>  
<Measure>  
<MeasureTypeCode>02</MeasureTypeCode>  
<Measurement>21</Measurement>  
<MeasureUnitCode>mm</MeasureUnitCode>  
</Measure>  
<Measure>  
<MeasureTypeCode>03</MeasureTypeCode>  
<Measurement>15</Measurement>
```

```
<MeasureUnitCode>mm</MeasureUnitCode>  
</Measure>  
<Measure>  
<MeasureTypeCode>08</MeasureTypeCode>  
<Measurement>1</Measurement>  
<MeasureUnitCode>gr</MeasureUnitCode>  
</Measure>  
<SupplyDetail>  
<SupplierName>Anko Warehousing</SupplierName>  
<TelephoneNumber>+44 1865 331995</TelephoneNumber>  
<FaxNumber>+44 1865 331996</FaxNumber>  
<EmailAddress>enquiries@anko.ie</EmailAddress>  
<ReturnsCodeType>03</ReturnsCodeType>  
<ReturnsCode>Y</ReturnsCode>  
<AvailabilityCode>IP</AvailabilityCode>  
<ProductAvailability>20</ProductAvailability>  
<ExpectedShipDate>20050520</ExpectedShipDate>  
<PackQuantity>50</PackQuantity>
```

```
<Price>  
<PriceTypeCode>01</PriceTypeCode>  
<DiscountPercent>10</DiscountPercent>  
<PriceAmount>34.99</PriceAmount>  
<CurrencyCode>AED</CurrencyCode>  
<CountryCode>AD</CountryCode>  
</Price>  
</SupplyDetail>  
<PromotionCampaign>Author tour and comprehensive point of sale material.</PromotionCampaign>  
<PromotionContact>Sara Elegerot  
sara@anko-publishing.com</PromotionContact>  
</Product>  
</ONIXMessage>
```