Data Recipient Best Practice

Field Dependencies and Troubleshooting

Prepared by Stephen Long
June 2025 Version 1.0

Disclaimer:

Please note: The information provided in this document is intended for guidance purposes only. Those involved in the creation, collection, management or distribution of product metadata are strongly advised to seek guidance on compliance with the business policies of their respective organisations.

Index

1. FIELD DEPENDENCIES

a. What are they?

An ONIX product record captures all the important information about a book product. A book's attributes are extensive, and looking at any of these in isolation can make them meaningless. For example, what use is an ISBN without a Contributor? A Title? A Product form? A supplier from which to obtain it?

What is required is the correct context. This comes from looking at the totality of the information for the product, or a subset of that information. Within the product record there are groups of related attributes which together help to provide a better understanding of the product.

For example, a publishing status of 'forthcoming' should be accompanied by an expected local publication date. Interpreting one or other piece of information, rather than both together, would be unhelpful. The assumption is that if the publishing status is 'forthcoming', then the expected local publication date will be in the future. Both together help the buyer or consumer make an informed choice about ordering. Together, these two separate attributes form a field dependency.

However, there is also a requirement that the dependency between the two attributes is logical. In the above example, the product with a publishing status of 'forthcoming' should have an expected local publication date that is in the future. A publication date that is in the past would not make sense. Either the publishing status or the expected local publication date would be incorrect in that scenario. Indeed, it indicates that the data as a whole is either old, or erroneous, and calls into question the reliability of the remainder of the data.

Whilst individual attributes may not be entirely independent of one another, so inevitably provide opportunities to create nonsensical or internally inconsistent data, they can also provide context and remove the risk of error or ambiguity.

b. Other examples of field dependencies include:

i. Measure type (weights and dimensions) and product form - for example, if the product form is 'e-book' and the measurements indicate that it has a weight and a size, then this is wrong.
ii. Contributor role (illustrated by) and Illustration, Other Content Type and Illustrated/ Not Illustrated
iii. Contributor role (Translated By) and Language Role (Original Language of a Translated Text)
iv. Sales Rights – for example, if the sales rights are WORLD, the REST OF WORLD sales rights must be omitted.

c. Managing Field Dependencies:

Field dependencies can vary according to business need, and whilst there is no exhaustive list, there are steps that senders and recipients can take to ensure that they are both logical and consistent when they occur:

i. An internal field dependency. Look for contradictions in data, where one attribute conflicts with another.
ii. An external field dependency. Check for data that could be open to interpretation by the data recipient.

For more information and related resources, please go to the EDItEUR website.

2. TROUBLESHOOTING

a. What do we mean by 'troubleshooting'?

This is simply another way of describing how to identify and resolve problems with metadata. Troubleshooting can apply to an ONIX feed, a message or an individual record. Not only that, but each situation can also be different, dependent on the assumptions and expectations of the sender and receiver of the metadata.

b. Types and Examples of Troubleshooting

There are three main categories of troubleshooting. Each is described below:

i. Validation of the ONIX XML for adherence to the message standard.

An XML schema formally defines the set of markup tags that may be used in a particular type of XML document, whether each tag is mandatory or optional, and their order and nesting. The schema also constrains the data types and values that may be used within the data elements in the document. It can require that a particular tag contains an integer, or a date, or set a limit on the length of text. An XML schema can define lists of allowed values ('enumerations', controlled vocabularies, or in ONIX terminology, code lists) that can be used in a particular data element. Validation is the process of checking that a particular XML file meets the requirements defined by a particular schema.

Note that there are different schemas available (the classic XSD, the strict XSD and RNG). Use of the DTD schema is not recommended.

ii. Incorrect data in the message.

This can be the result of business processes (including human error), or systems providing incorrect data (format, content) and can only be fixed after a thorough business process analysis that covers the lifecycle of the data itself. Who creates the data in the sender's internal system? Who is responsible for ensuring that updates to the data are applied? Are there checks – perhaps at key points in the lifecycle (e.g., at four months prior to publication and immediately prior to publication)? How is the data updated post-publication?

iii. Additional requirements.

This happens when the ONIX standard is adapted (usually by a large data recipient imposing its own 'data recipient guidelines') to cater for a specific business need, such as making a conditional data element mandatory.

Such additional requirements are not usually a problem, unless they impose a need to 'misuse' a particular field for a requirement other than it's originally intended purpose. This can conflict with other recipients' needs and inevitably reduces the interoperability of the ONIX data. This issue is often termed the 'flavours of ONIX'. Data recipients imposing additional requirements that demand a unique 'flavour' should be mindful of the extra 'cost to serve' that each 'flavour' imposes on their supply chain partners.

c. Tips for Troubleshooting

i. Regularly housekeep documented standards and ensure feeds reflect the current iterations of the format specification and code lists.
ii. When establishing new data feeds, ensure that test data is exchanged, validated and processed in advance of any 'go live'. Multiple exchanges may be required to finesse the feed and resolve issues. The strict XSD can be useful during the onboarding process with a new data sender or recipient (though note that you will need a full-featured XML software toolkit to use it, as the strict schema requires XSD 1.1 support). Oxygen is one suitable toolkit, but others are available.
iii. Where business processes and human intervention have a role to play in the creation, management and communication of metadata, ensure that data is regularly housekept (e.g., that SANs and GLNs are current and correctly formatted).
iv. Subscribe to the ONIX discussion forum at **groups.io/g/onix**
v. Familiarise yourself with other ONIX and metadata resources on the www.editeur.org and www.bic.org.uk websites.

EDItEUR's *Twelve ONIX Problems* and *Another Twelve ONIX Problems* highlight the common issues faced by ONIX senders and receivers. The Application Notes are available here: https://www.editeur.org/93/Release-3.0-and-3.1-Downloads/#HowTo