ONIX for Repertoire: message structure

Revised May 2010

CONTENTS LIST

Changes made May 2010

The document has been revised to coincide with the release of ONIX-DS Version 1.1 so that it no longer refers to “version 1.0”. Instead, it has been generalised to refer to “version *n.n*”, and in a number of other respects, so that it will not need to be updated for future versions of the ONIX-RP message unless there are other changes which affect the overall structure or the header and trailer examples.

1. Introduction

This document provides guidelines on the overall ONIX-RP message structure and details of the message header format.

The header details take the form of a table whose first three columns are a compressed form of the ONIX-RP Overview. Where possible, the fourth column shows typical content for each individual element, color-coded to indicate dictionary terms, labels and other data strings in the same way as in other ONIX documentation. (For composite elements, the column carries the word “*Composite*”, since in this case the content comprises the included elements.) The fifth column carries application notes and business rules.

Following the guidelines, examples are given of PLS-CLA and CLA-CCC XML message headers.

2. Message structure

The overall structure of an ONIX-RP message is illustrated on the next page, and has the following components:

XML declarations

Each message starts with a line of “XML declarations”, which are explained in section 3.

Start of message

The second line announces the start of an ONIX-RP message, declares which version of the ONIX-RP format is being used and states the location of the ONIX-RP XML Schema for the specified format version, against which the message should be valid when checked by an XML schema parser (XML syntax checking tool) – see section 4 for more detail.

Message header

The first element of the message is a mandatory header, identifying the sender and addressee of the message with some additional “housekeeping” information.

Repertoire notifications

The body of the message consists of an indefinite number of RepertoireNotification composites, of the same or different types. The order of such elements in the message is not significant.

End of message

The message ends with a single line **`</ONIXRepertoireMessage>`**

```
<?xml version="1.0" encoding="UTF-8"?>
<ONIXRepertoireMessage version="n.n">
     <Header>
          Message header
     </Header>
     <RepertoireNotification>
          Repertoire notification 1
     </RepertoireNotification>
     <RepertoireNotification>
          Repertoire notification 2
     </RepertoireNotification>
     <RepertoireNotification>
          ......................
          ......................
     </RepertoireNotification>
</ONIXRepertoireMessage>
```

3. XML declarations

The XML declarations line at the start of the file simply alerts any receiving system to the fact that this is an XML file and that the character encoding used in this file is Unicode (UTF-8). Although it is not obligatory in general for an XML file to contain an XML declarations line, it is a requirement for the ONIX-RP format that it be included, to avoid uncertainty as to the intended format and encoding of an ONIX-RP message. Adoption of UTF-8 is very strongly recommended since (a) it is the default encoding for any XML string, and (b) alternative encodings frequently create problems with the representation of extended character sets.

If the file is generated using software tools that are "XML-aware", the chances are that this line will be inserted automatically and with the correct information. However, if the file is generated using tools that are not "XML-aware" – for example, by a conventional report generator on a database management system – care must be taken to ensure that this line is correct in every detail, and that the character encoding of the file is indeed Unicode and not some other encoding (eg the Windows character set, which is the same as ISO 8859-1). Developers of systems for generating ONIX-RP messages are advised to pay particular attention to this point.

4. Start of message

The second line contains the XML "tag" that marks the start of the message. This is not the place for a detailed explanation of XML syntax, but as this line contains further vital "XML declarations", some description is necessary:

< marks the beginning of the XML tag.

`ONIXRepertoireMessage` is the name of the top-level XML “element” and must immediately follow **<** without any intervening space or other characters.

A mandatory space follows the name of the top-level element.

`version="`*`n.n`*`"` is a mandatory XML “attribute” that specifies the ONIX-RP message format version number.

> marks the end of the XML tag, and may be preceded by a space.

5. Header guidelines

Each ONIX-RP message carries a simple header, which must as a minimum (a) identify the sender and addressee, (b) carry a sequential message number, (c) show the date and (optionally) the time when the message was sent, and (e) specify a default currency for monetary amounts included in the message. Other content is optional. The header format is common to many current ONIX applications.

	ONIX element	ONIX description		Typical content	Application notes
1	**<Header>**	**Message header**	**1**	*Composite*	
2	<Sender>	The sender of the message (coded identifier or name or both)	1	*Composite*	
3	<SenderRole>	A controlled value specifying the role of the sender if not the delegator of rights	0-1		
4	<SenderIdentifier>	Composite: a coded identifier of the message sender, eg a SAN or GLN	0-n	*Composite*	Coded identifiers are optional in this application
5	<SenderIDType>	An ONIX controlled value specifying a scheme from which an identifier is taken	1		
6	<IDTypeName>	A name of a proprietary scheme, if applicable	0-1		Not used
7	<IDValue>	An identifier value, from the specified scheme	1		
8	<SenderName>	The name of the sender organization	0-1	PLS	Mandatory in this use case
9	<SenderContact>	The name of a contact person in the sender organization	0-1		
10	<SenderEmail>	An email address for the sender	0-1		
11	<Addressee>	The addressee of the message (coded identifier or name or both)	1-n	*Composite*	Mandatory
12	<AddresseeIdentifier>	Composite: a coded identifier of the message addressee	0-n	*Composite*	Coded identifiers are optional in this application
13	<AddresseeIDType>	An ONIX controlled value specifying a scheme from which an identifier is taken	1		
14	<IDTypeName>	A name of a proprietary scheme, if applicable	0-1		
15	<IDValue>	An identifier value, from the specified scheme	1		
16	<AddresseeName>	The name of the addressee organization	0-1	CLA	Mandatory in this use case
17	<AddresseeContact>	The name of a contact person in the addressee organization	0-1		Not used
18	<AddresseeEmail>	An email address for the addressee	0-1		Not used

	ONIX element	ONIX description		Typical content	Application notes
19	<Test/>	An XML "empty element" which, if present, specifies that the message is a test	0-1		
20	<MessageNumber>	Message sequence number	0-1	24710	Increments by 1 for each successive ONIX-RP message sent by PLS to CLA.
21	<MessageRepeat>	A number which distinguishes any repeat transmissions of a message	0-1	0	Zero the first time a message is sent; increments by one each time it is necessary to re-send the same message, eg because of a transmission failure.
22	<SentDateTime>	The date, and optionally the time, when a message was sent. Permitted date and time formats are as follows (letters T and Z represent themselves): YYYYMMDD Date only, eg 20070423 YYYYMMDDTHHMMSS Date and time, eg 20070423T212034 YYYYMMDDTHHMMSSZ+HHMM YYYYMMDDTHHMMSSZ-HHMM Date, time and time zone, eg 20070423T212034Z-0500	1	20100516T140345	Date and time without time zone
23	<MessageNote>	A free-text note about the contents of the message	0-1		Not used
24	<CurrencyCode>	An ISO code specifying the default currency for monetary amounts in the message	0-1	iso:GBP	Required if the message is carrying prices

6. Header examples

PLS to CLA

```xml
<Header>
        <Sender>
                <SenderName>PLS</SenderName>
        </Sender>
        <Addressee>
                <AddresseeName>CLA</AddresseeName>
        </Addressee>
        <MessageNumber>5763</MessageNumber>
        <MessageRepeat>0</MessageRepeat>
        <SentDateTime>20070516T140345</SentDateTime>
        <CurrencyCode>iso:GBP</CurrencyCode>
</Header>
```

CLA to CCC

```xml
<Header>
        <Sender>
                <SenderName>CLA</SenderName>
        </Sender>
        <Addressee>
                <AddresseeName>CCC</AddresseeName>
        </Addressee>
        <MessageNumber>1562</MessageNumber>
        <MessageRepeat>0</MessageRepeat>
        <SentDateTime>20070717T100536</SentDateTime>
        <CurrencyCode>iso:USD</CurrencyCode>
</Header>
```