ICT-PSP Project no. 270905
LINKED HERITAGE
Coordination of standard and technologies
for the enrichment of Europeana

Starting date: 1st April 2011
Ending date: 30th September 2013

Deliverable Number:	D 4.2
Title of the Deliverable:	Specification of the technologies chosen
Contractual Date of Delivery to EC:	Month 18
Actual Date of Delivery to EC:	October 2012

Project Co-ordinator

Company name :	Istituto Centrale per il Catalogo Unico (ICCU)
Name of representative :	Rosa Caffo
Address :	Viale Castro Pretorio 105, I-00185 Roma
Phone number :	+39.06.49210427
Fax number :	+39.06. 06 4959302
E-mail :	rcaffo@beniculturali.it
Project WEB site address :	http://www.dc-net.org

Context

WP 4	Public private partnership
WP Leader	Michael Hopwood (EDItEUR)
Task 4.3	Metadata model selection
Task 4.4	Technical specification
Task Leader	Michael Hopwood (EDItEUR)
Dependencies	D4.1

Author(s)	Michael Hopwood (EDItEUR)
Contributor(s)	
Reviewers	Hugo Manguinhas, IST
Approved by:	

History

Version	**Date**	**Author**	**Comments**
0.1	21 Sept 2012	Michael Hopwood	
0.2	05 October 2012	Michael Hopwood	After internal review by EDItEUR staff and WG4 partners
1.0	13 October 2012	Michael Hopwood	Revision as per review by IST

CONTENTS

1 EXECUTIVE SUMMARY

The European Commission's "Comité des Sages" report on digitisation, "The New Renaissance", expressed the desirability of enabling online discovery of contemporary, born-digital and digitised cultural works, which, because they are currently in-copyright (and may be in-commerce), have not been digitised, and so are often invisible online; the so-called "20th Century Black Hole". Addressing this need, Work Group 4 of Linked Heritage (linkedheritage.eu) aims to specify how metadata describing relevant commercial products in four media – books, music recordings, film and TV, and photographs – can be aggregated and integrated with cultural heritage data in portals like Europeana (europeana.eu).

WG4'sfirst deliverable (D4.1 Best Practice Report – Public-Private Partnership; available at linkedheritage.org/getFile.php?id=283) described the benefits of this approach to both the heritage and commercial sectors, in broad outline how it could be achieved (both technically and in practical, legal-commercial terms), and laying a foundation for detailed plans by describing the most prominent industry standards in each of four media sectors.

The current report, D4.2 Specification of Technologies Chosen, builds on the findings of D4.1 with the results of Tasks T4.3, an empirical estimate and evaluation of potential commercial data contributors, and T4.4, an evidence-based technical specification for aggregating such data at scale. The current state of theory and practice in data integration is reviewed, focussing on efforts to achieve cultural-commercial sector interoperability, and potential solutions for the problem at hand are considered for feasibility given the limited resources. An experiment was undertaken to assess the feasibility of applying Linked Heritage's existing data integration format, LIDO, and aggregation platform, MINT, to the previously identified industry standard metadata formats. The experiment's focus was the ONIX for Books 3.0 mapping described explicitly in Linked Heritage's Description of Work, although DDex for recorded music, EIDR for audiovisual materials and IPTC for photos were also investigated. This was done with the knowledge and cooperation of the relevant standards bodies, to achieve a reliable and standardised result in accordance with accepted industry best practice.

The report finds that, although other technical solutions exist – and some have been applied successfully to integration of commercial and heritage data – Linked Heritage's existing pragmatic solution is adequate to this task in the case of ONIX for Books 3.0 and shows promising signs for the other three schemas. There is basic semantic compatibility in practice, confirming the theoretical assumptions of Linked Heritage D4.1. The existing ONIX mapping can be further tested and refined in support of discussions with potential commercial sector data contributors, and experiments on their test and prototype data, as work towards D4.3.

In order to progress from semantic schema mappings to a full-scale aggregation of data, significant technical questions remain to be answered in the cases of IPTC and EIDR data, and several enhancements to the LIDO schema and to the MINT aggregation software are proposed to bring Linked Heritage's aggregation model in line with current commercial metadata best practice, as exemplified in the schema mappings considered.

Sources of data for each standard are described according to the likely amounts and quality of data available, and the costs, legal framework and technical requirements for accessing them. These themes will be expanded upon in the remaining deliverable from this Work Package.

Finally, the report recommends specific work to assist these enhancements, both within Linked Heritage and also the wider cultural heritage community, including Europeana itself and the international cultural heritage documentation committee, CIDOC (network.icom.museum/cidoc/).

2 INTRODUCTION

"That belongs in a museum."

- Henry Jones Junior, in *Indiana Jones and the Last Crusade,* 1989

"Yon second-hand bookseller is second to none in the worth of the treasures which he dispenses."

Leigh Hunt, *On the Beneficence of Bookstalls*[1].

The definition of cultural heritage[2], that which belongs in museums, galleries, libraries and archives, is officially framed in general, abstract terms. One might expect digital libraries, and especially those on a national or EU-wide scale, to follow this familiar, somewhat academic route. But Europeana[3] and the European Commission's Comité des Sages' report[4] have taken a pragmatic approach as to what should be visible and accessible to European citizens, both in the heritage sector and the commercial cultural industries, and this goes far beyond the traditional categories of unique items witnessing to historic or culturally formative events, to encompass the industrially mass-produced products of contemporary cultural industries: books, recorded music, film and TV, and photographs.

The previous deliverable of Linked Heritage Work Package 4 described the metadata available in the commercial cultural industries; this report documents work done to enable that metadata to be integrated with the existing cultural heritage corpus.

2.1 BACKGROUND TO THIS DELIVERABLE

The desire to integrate information and the metadata describing it across multiple domains is a less recent phenomenon than it might first appear, going back perhaps at least as far as the 19th Century explosion of publication and the "documentalists" who strived to organise it (van den Heuvel & Rayward, 2011). In fact, the growing sense of "information overload" and the need to develop tools for managing and navigating the "deluge" means that information integration is at least implicit in most modern library and information work, as well as becoming a key component in commercial enterprise data management.

Another, complementary motivation may be the sense that, for the first time, through new networked technology, and convergence between theory and practice across media and disciplines, it may be possible to gain an overview of previously scattered cultural information; the experience of the "grand tour" through European history and culture but at the level of fine detail, sharing some of the intimacy of the painstaking curators and students of artefacts and ideas, without all of the normally requisite years of preparation.

2.2 AIMS OF THIS DELIVERABLE

This deliverable reports the fulfilment by Linked Heritage Work Group 4 of Tasks T4.3 and T4.4, as well as some of the more general objectives, relevant to the Tasks. WG4 fulfilled these tasks by:

- Communicating the scope of the problem of integrating commercial sector metadata with Europeana (and cultural heritage data generally);
- Surveying and selecting among the available approaches with Europeana, other similar projects and the professional and academic literatures;

[1] Quoted in *Hoyt's New Cyclopedia Of Practical Quotations*, 1922, p. 649.
[2] See Linked Heritage, Deliverable D4.1, Appendix 2 – Glossary of Terms
[3] See, for example, Europeana's collection development policy [http://pro.europeana.eu/documents/866205/0/EV1-AF-ContentDevStrategy.pdf] and 2011 annual report [http://pro.europeana.eu/documents/858566/ade92d1f-e15e-4906-97db-16216f82c8a6]
[4] See http://dx.doi.org/10.2759/45571

- Producing and describing demonstrators for some approaches, evaluating these for effectiveness in integrating commercial sector data into Europeana;
- Proposing practical ways to address the opportunities and challenges for public-private partnerships with Europeana in future.

Drawing on the Linked Heritage Description of Work (DoW), the Work Group understood its tasks leading to this deliverable as follows.

2.2.1 Task T4.3 - Metadata model selection

"The third task will be to assess the various knowledge resources identified above (T4.1) and to select the metadata model which offers the best potential for sizeable contributions to Europeana by the private sector. Selection criteria will include

- Established user base;
- Adherence to standards and/or standards status in its own right;
- Demonstrated interoperability with other metadata models, including those familiar to the public sector;
- Demonstrated and/or potential ease of integration with the technologies selected in other thematic work-packages (i.e. Linked Data, PID, selected metadata models);
- Maturity and quality of available technical implementation, documentation and support."

For this task, "the various knowledge resources identified above" was understood to mean the standard identifiers, metadata schemas and related services described in D4.1, Best Practice Report – Public-Private Partnership, inasmuch as they are used in actual practice to create corpora of data. In this light, "the metadata model which offers the best potential" will be the most promising model from each sector, making a total of four models selected.

The above selection criteria were accepted as helpful and important by the Work Group, and expanded to include five others of importance technically and with the final WP4 deliverable in view:

- Technical access to data;
- Legal access to data;
- Cost of access to data;
- Potential to enrich metadata content;
- Links into existing cultural heritage metadata corpus.

Simply put, these criteria will form the basis of a cost-benefit analysis of potential services to deliver the integrated data.

2.2.2 Task T4.4 - Technical Specification

"The fourth task of this work-package will be to specify the technical components of the large scale implementation (validation) platform (see WP5) which are concerned with ingestion of private sector content into Europeana, including

- The metadata models used;
- Mapping these metadata models to ESE/EDM (possibly using an interim metadata model)."

This task constituted the empirical work of ingesting samples of commercial data into the MINT aggregator and attempting to create mappings of the source schema's semantics to the LIDO schema, to discover how far such mapping is possible, given the different objects of interest of the source and target descriptions (unique items *versus* classes of products – see D4.1, section 4.3. and this report, section 18). It also encompasses reviewing the capabilities of MINT to deal with the requirements of commercial data for updates and on-going data management (see D4.1, section 5.3.5.).

2.3 SCOPE OF THE DELIVERABLE

The above general aims and tasks include some that seem at face value extremely far-reaching; for example, estimating how much product data is available from the entire EC's creative industries, mapping extremely detailed data schemas to the (in principle) indefinitely extensible LIDO schema, and specifying a workable technical model (or models!) for a production-scale commercial data aggregation service.

However, partly through further reflection on the issues raised in D4.1, and partly due to detailed knowledge of the metadata schemas mapped for this deliverable, it was understood that in order to make progress, the aims had to be operationalised in concrete, limited and extremely focussed ways.

Therefore, although the full spectrum of best practice, possible experimental approaches, and actual technical work is considered here and described as fully as possible, the report describes only the ONIX mapping required by the Description of Work in detail, and in order to provide maximum value and set a milestone for best practice, provides outlines and high-level specifications for mappings in the other domains.

2.4 CURRENTLY AVAILABLE SUPPLEMENTS TO THIS DELIVERABLE

Full exploration of the domains to be integrated produced extensive and potentially useful results beyond the expected outcomes. Several supplements to this deliverable are available upon request from EDItEUR:

1. ONIX for Books 2.1 full mapping documentation (spreadsheet);
    a. IPTC Core and Extension: semantic mapping document (spreadsheet);
2. EIDR:
    a. Semantic mapping document (spreadsheet);
    b. MINT mapping in XSLT;
3. DDex:
    a. Semantic mapping document (spreadsheet);
    b. MINT mapping in XSLT.

In addition, advice and assistance in using the mappings for a schema can be offered depending on interest and collaboration from potential data providers using that schema to support the testing and prototyping work of D4.3.

Note that these documents are not part of the current deliverable; they represent value added by Work Group 4 partners during the creation of D4.2, above and beyond the Description of Work. They are offered to support the promotion of Linked Heritage's objectives through the Best Practice Network, especially in demonstrating the project's value to commercial players in the photo, music and AV domains, and mature versions may be uploaded to the project website or published as part of D4.3.

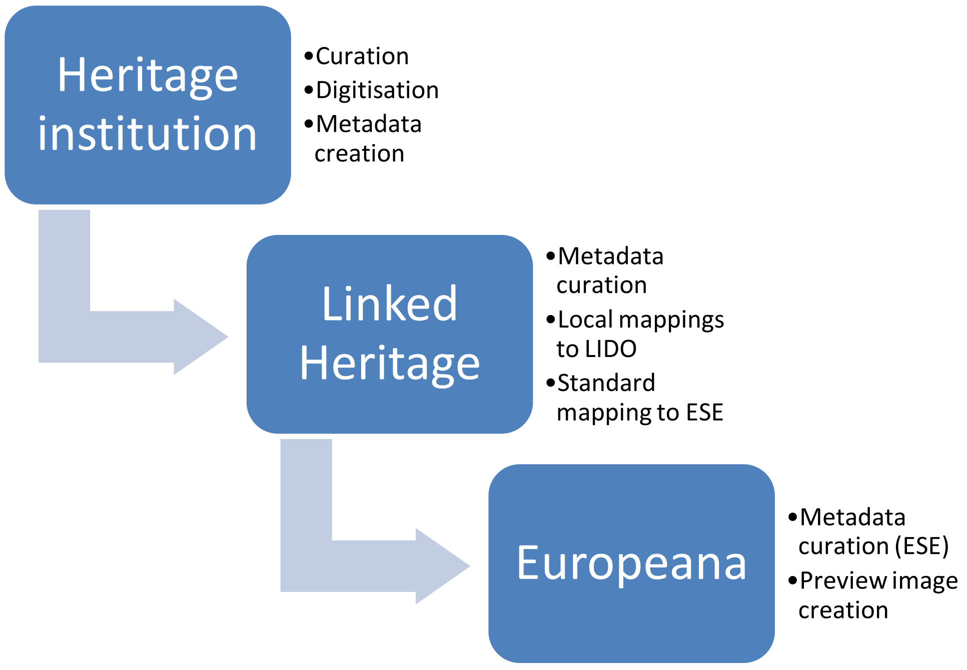
3 RESEARCH AND EXPERIMENTS UNDERTAKEN

The first deliverable (D4.1) of Work Package 4 took an expertise-gathering approach relying on expert contacts in the relevant content industries and standards communities, synthesising and clarifying the best practice across sectors. This deliverable describes practical experiments in data mapping and integration, although of course based on the standards and best practice described in D4.1, with a further literature review and continued advice and assistance from the network of interested experts, primarily within the standards bodies themselves, but also, especially in the case of ONIX for Books and IPTC, members of the standard user communities.

3.1 RESEARCH CONTEXT

The Linked Heritage consortium, continuing the work of the earlier ATHENA and MINERVA projects, has developed a well understood and tested standard method for aggregating cultural heritage data for preservation, standards development and experimentation, and contribution to Europeana. This forms the context and starting point for the current work. It is useful to review the existing process for its role in the method and findings of the experiments and as a benchmark for comparison with the proposed solutions.

Linked Heritage, much like other Europeana Network projects, acts as aggregator, coordinator and metadata gateway for its partners' contributions to Europeana[5] as outlined below:



The first step of the Linked Heritage process constitutes creation of textual metadata describing the cultural heritage object, and usually linking them to associated digitised surrogates, such as photographs of historical artefacts, scans of manuscripts, sound samples from recordings, or digitised AV, accessible somehow on the institutions' websites. This is valuable and highly authoritative data since it is produced by the object's curating institution, but its format may be more or less standardised. The ATHENA survey on existing standards applied by European museums (and other heritage institutions)[6] found that out of 133 respondent institutions, 23 used idiosyncratic local data formats, a number significantly higher than for any single standard format. The most commonly used standard was Dublin Core[7] (22 institutions), which is usually substantially altered for

[5] A more detailed version of this diagram is found in Appendix 4.
[6] ATHENA Deliverable D3.1 – available at: http://www.athenaeurope.org/index.php?en/149/athena-deliverables-and-documents
[7] The survey did not record the serialisation and/or data model, but given the experience of Europeana and ATHENA, probably it was some type of flat XML structure rather than an RDF representation.

each local use[8] and in effect adds to the "local standards" number, making it more significant. The need for a central, broadly-applicable aggregating standard to avoid mapping an extra 43 "standards" to Europeana's schema is clear. In any case, once the data describing heritage objects and collections is identified and approved for contribution to the project, it can be aggregated in the second step.

A large part of the value added by Linked Heritage is at this second, intermediating stage, since after upload, it is normalised by a semantic mapping to the LIDO data harvesting schema[9]. This standard format solves, at least for the cultural heritage domain, the critical barriers to interoperability and useful search endemic to schemas like Dublin Core[10]. Because LIDO is based on the most comprehensive and widely adopted existing schemas, and mostly adopts their definitions, it is appropriate for the domain-specific data of most contributors. Its major strength as an aggregation format, though, is in its harmonisation with the CIDOC-CRM[11], which makes its structure more flexible and extensible, by generalising most of its conceptual categories and explicitly filling in the relationships between them, which are usually implicit in data schemas, and often ambiguous or non-existent in schemas like Dublin Core. Because Linked Heritage transforms all metadata contributions to LIDO, it creates an immense resource of rich, interoperable data that can be of value to the contributing institutions and the heritage sector more generally. To portals such as Europeana, it is a more helpful long-term content provider, because having normalised all source datasets, it is able to provide one standard mapping to the vastly simpler, "dumbed-down" Europeana Semantic Elements schema (or ESE, a Dublin Core "application profile"), and adapt this one mapping whenever the Europeana schema changes.

The final step of publication to Europeana (or some other portal, or data endpoint, potentially) can thus be managed centrally, but with a fine control and agility impossible for heritage institutions concerned with their "business as usual" and of course other projects. This has been demonstrated, for example, in the response of Linked Heritage to the recent introduction of the requirement[12] for all Europeana contributors to sign a CC0 waiver[13] of all current and future rights in their data; as it may not be possible, or desirable, for many Linked Heritage partners to release their entire corpus of data this way, a "filtering" option has been introduced into the aggregation server to allow fine control of the level of detail published. At this point it should also be noted that the stated rationale[14] for Europeana's move to CC0 for textual metadata is to enable publication of Linked Open Data[15]. The proposed model for this, already tested in Europeana's pilot dataset publication in 2011[16], is Europeana Data Model (EDM), essentially consisting of a somewhat extended version of ESE[17], and thus only

[8] So-called "application profiles" of Dublin Core for any serious use tend to require inclusion of elements entirely foreign to the Dublin Core namespace; in other words, they are in fact distinct new standards which happen to include DC elements. See, for example, the Scholarly Works Application Profile for academic journal eprints [http://www.ukoln.ac.uk/repositories/digirep/index/Scholarly_Works_Application_Profile] or the MICHAEL-EU profile for heritage collection descriptions [http://www.ukoln.ac.uk/metadata/michael/michael-eu/dcap/]

[9] www.lido-schema.org/

[10] See ATHENA Deliverable D3.2, section 3.3 for a full discussion of the inadequacy of Dublin Core to even simple searches over rich data, and how LIDO demonstrates significant improvement in this and other respects [http://www.athenaeurope.org/index.php?en/149/athena-deliverables-and-documents]. Note also that Dublin Core Metadata Element set *per se* is not a schema, so in practice every "application profile" developer is required not only to rethink the semantics but also the syntax of their implementation of DC.

[11] See explanation of LIDO's basis in existing standards and CIDOC-CRM harmonisation at http://network.icom.museum/cidoc/working-groups/data-harvesting-and-interchange/lido-overview/related-standards/

[12] The new Europeana "Data Exchange Agreement" – see http://pro.europeana.eu/web/guest/support-for-open-data/faqs

[13] See "About CC0" at http://creativecommons.org/about/cc0

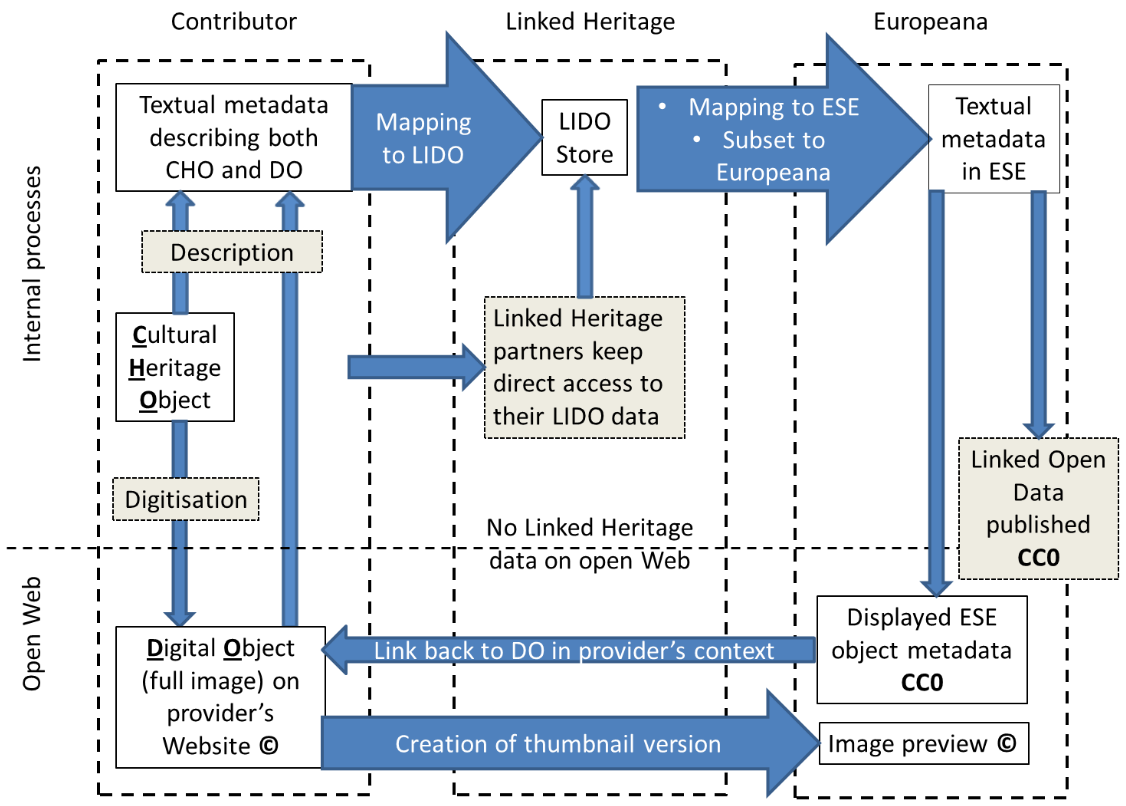
[14] See "Support for Open Data at http://pro.europeana.eu/web/guest/support-for-open-data

[15] See the W3C pages on Linked Data for details of the connection between "linking" and "opening" data: http://www.w3.org/standards/semanticweb/data

[16] Described in a paper at http://dcevents.dublincore.org/index.php/IntConf/dc-2011/paper/view/55

[17] EDM is fully described here: http://pro.europeana.eu/web/guest/edm-documentation - although it represents significant progress beyond the "flat" ESE schema consisting mainly of the Dublin Core (qualified) Metadata Element set, the fact remains that the added terms largely address the problems which arose from representing a complex aggregation workflow (see the full diagram below in this section, 3.1) in a simple,

superficially more robust and flexible than Dublin Core. The lack of a Europeana-wide normalisation pipeline for the core EDM data, and the existence of more detailed and explicit relationships in LIDO mean that LIDO is probably better positioned for direct production of Linked Data because that relies on the capacity for decomposing schematised data into atomic, unambiguous and reliable statements (“triples”)[18]. The full situation is summarised by this diagram of the data flows between the heritage organisation, Linked Heritage and Europeana, internally and onto the open Web:


Contributor
Linked Heritage
Europeana
Internal processes
Open Web
Textual metadata describing both CHO and DO
Mapping to LIDO
LIDO Store
Mapping to ESE
Subset to Europeana
Textual metadata in ESE
Description
Cultural Heritage Object
Linked Heritage partners keep direct access to their LIDO data
Linked Open Data published CC0
Digitisation
No Linked Heritage data on open Web
Digital Object (full image) on provider's Website ©
Link back to DO in provider's context
Displayed ESE object metadata CC0
Creation of thumbnail version
Image preview ©

Full aggregation workflow for Europeana and Linked Heritage

Key terms identified in the above diagram are:

- “Cultural heritage object” (CHO), the object of interest and value to be described. It may or may not be “born-digital” but digitisation at some stage is mandatory for inclusion in the workflow.
- “Digital object” (DO), the highest quality image(s) produced by digitisation of the original CHO. This is normally displayed on the open Web at the contributor’s Web site, together with some contextual information, including how to access the CHOand the rights associated with that access.
- Textual metadata, derived from the LIDO data stored in Linked Heritage’s servers.
- Image previews, derived from the DO, keeping all the same image rights as the DO.
- Links back to the DO in context, implying that the context for viewing the DO and its relationship to the CHO is controlled exclusively by the contributor.

---

resource-based format. Altering the entity assignments of the Dublin Core properties and adding some relations relevant to the heritage domain’s best practice are positive steps but still do not change the need to further specify the semantics of the (unchanged) DC properties themselves.

[18] See for example the initial investigation into creating linked data from LIDO by several Linked Heritage partners (Tsalapati *et al.*, 2012), at http://www.cidoc2012.fi/en/File/1663/simou.pdf - this also explains some of the limitations of EDM as compared with LIDO, the domain-specific heritage aggregation schema. EDM is specifically intended to produce linked data, but, also as a consequence of the problems noted in the footnote above, “…the quality of Linked data implementations is only as good as the data you are linking to, and the meaning and contextualisation of the link you use” (see http://www.doi.org/doi_handbook/5_Applications.html#5.4) – these essential quality issues are much more fully addressing in LIDO than in EDM.

Note also in this diagram that LIDO data is never directly exposed to the open Web.

3.2 METHODS AND APPROACH

Whereas the first deliverable of Linked Heritage synthesised the existing knowledge on standards and best practice in the cultural heritage and commercial metadata communities, and thus defined the terms of the problem of integrating them in broad terms, it was not able to make progress in providing solutions, beyond identifying two key areas which provide proof-of-concept:

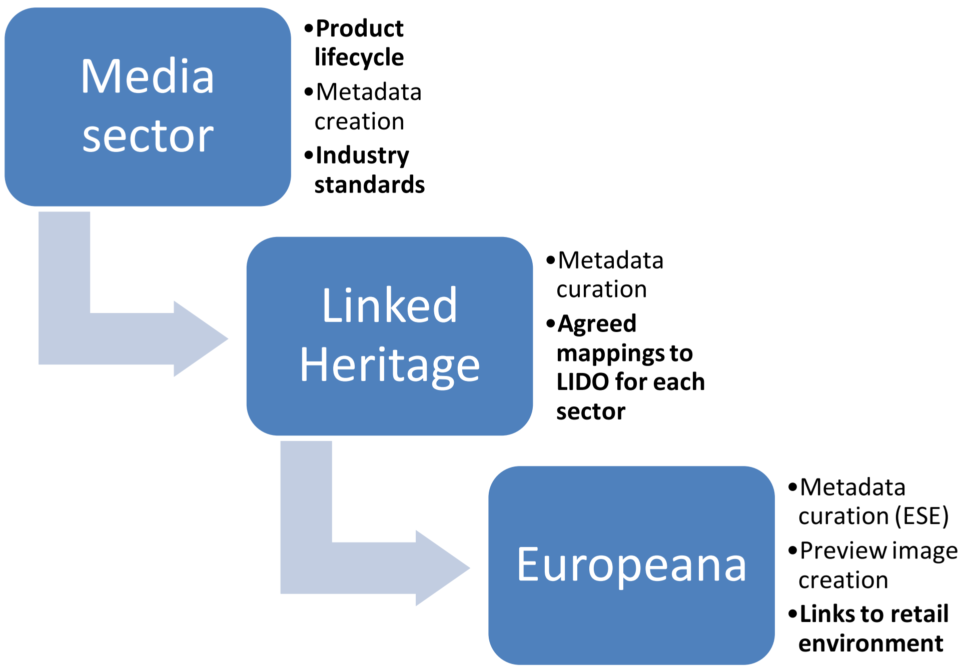
a) The existence of some projects and services, so far exclusively in the books and audio-visual domains, which embody a public-private partnership basis for data integration of commercial products (mostly, but not always, together with heritage counterparts) for discovery and links to access;
b) Conceptual and practical inroads into standards-based heritage-commercial sector interoperability (mainly but not exclusively in the books domain).

The examples identified in point a) demonstrate that the work in the current deliverable can be worthwhile, as at least in some cases, a level of financial and institutional commitment, and willingness to collaborate across sectors has moved beyond mere discussion. The partnerships in a), which were described in D4.1, section 8, were restricted to a single media sector each, but the aim of this research is to specify how to build on the more comprehensive tools developed in point b), as outlined in D4.1, section 5.4., to scale-up the basic partnership model of "culturally-relevant metadata for potential sales via retail links", across all four sectors, and preserving the maximum data richness.

As an empirical attempt to verify exactly this combination, the current research is relatively unprecedented. Previous attempts at all-round coverage have been a very high level of abstraction, which may not be suitable for practical, day-to-day data exchange[19]. Innumerable examples of full-schema mappings, including for commercial schemas like ONIX[20] have aimed at one-to-one compatibility with another specialised schema, rather than, as LIDO does, explicitly enabling re-use outside the immediate domain of interest of the source schemas.

This work is best understood as a first investigation to determine the precise extent of progress in practical semantic interoperability between the whole cultural heritage and commercial sectors, whose results will include practical advice for the short term integration of information, recommendations for both sectors, and specifications for new tools or revisions to existing standards to implement the known best practice. As noted above in 2.3., the practical work strove to find a very practical balance between the ideal solution and the already existing, more or less *ad hoc* compromises. This was based on following the Linked Heritage model as explained in the previous section (3.1.) with three important modifications:

[19] See Stein, *et al.* (2005).
[20] See http://www.loc.gov/marc/onix2marc.html and http://www.editeur.org/96/ONIX-and-MARC21/



The most salient difference in aggregating commercial product data, in contrast to heritage object data, is that we are concerned with the "lifecycle" of a generic, mass-produced product as it passes between various partners in a supply chain, in contrast with the "life history" of a unique artefact or found object in the museum world, considered as "an illustration, or witness of the past"[21]. A full discussion of the contrast is found in our previous report, D4.1., sections 5.1. to 5.3. As before, metadata creation is part of the first stage, but rather than taking an individual commercial cultural organisation (*e.g.* a single book publisher, record distributor, film company or photo library) as the starting point and expecting to map many local schemas to LIDO, we take the sector as a whole, represented by the relevant industry data standard (full descriptions are found in the relevant sections of D4.1):

Media industry sector	Product data standard	D4.1 section:
Book publishing	ONIX for Books	6.3
Recorded music	DDex ERN	6.4
Film and TV	EIDR / ISAN	6.5
Photography	IPTC Core and Extension / XMP	6.6

The use of relevant sector standards should have several benefits for contributors and for Linked Heritage / Europeana:

- Existing companies that use their industry standards can most easily and effectively contribute data as Linked Heritage partners, and rely on the proven suitability of the standard data format to express the creative integrity and legal-commercial identity of their products; others can adopt the standard, possibly with support from the relevant experts, and gain the associated benefits of efficiency and savings in data exchange IT, potential improvements in local systems design, ability to exchange product information with a wider range of partners, and so on;
- The heritage sector can expect richer and better structured data, probably with more inherent cultural value, and certainly more robust design, making it more suitable for data integration and linked data applications.

As will be explained below in section 4.4., the best practice for creating a semantic mapping between two existing standards is to create an agreed statement of equivalence that is itself "standardised", in the sense of resulting from understanding and authorisation of all parties involved (at minimum, the maintaining bodies of

[21] Doerr, M. (2010) Technological Choices of the ResearchSpace Project. Available at: http://www.researchspace.org/researchspace-concepts/technological-choices-of-the-researchspace-project

the two standards mapped). This is the key difference at the Linked Heritage stage of the hypothetical workflow, and the point where EDItEUR and the other Work Group 4 partners add the most value, EDItEUR being one of the sector standards bodies itself, and having a successful history of involvement in interoperability work of this kind; EDItEUR, MVB and mEDRA providing governance and (for MVB and mEDRA) registration of persistent identifers.

As explained in D4.1., sections 4.3. and 5.4., the library sector, is a convenient intersection point where the object of interest for identification and description, and the methods for describing it, overlap in terms of uniqueness and context, since here commercially published, mass-produced books are documented in ways that often tend to the more purely historical approach of museums proper. Hence there is existing work to build on and considerable expertise to draw on. As demonstrated in Appendix 3, the modelling approach developed for library data is applicable across all media sectors.

Finally, as before, the LIDO dataset, or a subset thereof (divided either by records, fields, or both), may be contributed to Europeana via an appropriate mapping to ESE (soon to be superseded by EDM). At this level two aspects are essential to the legal and commercial viability of the whole "pipeline", since they are intrinsic to the reason for creating and sharing the product data at all:

- Inclusion of links to at least one source (this could be the producer or publisher) for the product;
- Acceptable selection and arrangement of data elements for display to potential buyers.

Providing retail links *per se* is technically relatively straightforward; selection and maintenance of appropriate links is a far more challenging problem, technically and commercially. Similarly, although technical solutions for mapping LIDO to ESE (and by virtue of its similarity, EDM) already exist or can be envisaged, the loss of detail and flexibility in the transition to ESE/EDM cannot be addressed only by technical means[22].

Because so much depends on tailoring the ESE/EDM terms to the local use (as explained in section 3.1) this is more a matter of considering the commercial needs informing customer-facing display within Europeana rather than concern for retaining maximum semantic value. Therefore these aspects of the problem have been investigated during the work on Tasks T4.3 and T4.4 but full discussion will be provided in the final deliverable of Work Package 4, D4.3 Specification of legal/licensing environment.

Taking all this into account, the following literature review thus covers the full range of academic and business research, several types of tools, standards and systems, and the outcomes of projects and standardisation efforts.

3.3 NOTE ON PRESENTATION OF TERMS AND SYNTAX

Throughout this report technical terms and syntactic symbols (mainly from XML) are used within the narrative text. Therefore they have been presented in a variety of forms suitable to reading; terms with specific definitions are always written in Title Case; terms taken from an XML schema in the case used in that schema (for example, CamelCaps or lowerCamelCase) and XML elements themselves written with <angle_brackets> and in a `10pt fixed-width font` when quoted from a piece of XML or XSLT. Terms are often presented with a prefix as in *namespace:term* to avoid confusion when two or more schemas are discussed together. Finally, an XPATH is sometimes presented truncated to the last few elements, when the root path is clear from the context of the discussion. The full XPATHs are always available in the mapping documentation provided in this report and its appendices and attachments.

[22] Other than enhancing the ESE and EDM models themselves.

4 LITERATURE REVIEW

As in the previous deliverable, a mixed research process informed the compilation of this report and the context for the technical decisions taken in the practical work. Just as in D4.1, the approach of the literature review is not academic but technical and results- and standards-oriented in its description of state-of-the-art, delineation of approaches, and selection of suitable methods.

4.1 LITERATURE SEARCHES

The same library science journals and journal collections were consulted as for D4.1[23] as well as a selection of Web searches focussed on the ac.uk and .edu domains. Typical search terms included "metadata schema mapping", "semantic mapping" and "data integration"[24]. The use of related terms with slightly different ranges of meaning was useful in giving historical depth on pre-Internet work on database integration and context beyond the commercial and heritage sectors. This is reflected in the two bibliographic lists at the end of this report, which include citations from the text but also indications for useful research beyond the immediate topics.

4.2 BEST PRACTICE REPORT – PUBLIC PRIVATE PARTNERSHIP

The first deliverable of Linked Heritage Work Package 4 had already been substantially completed and submitted for review when work started on metadata models, technical specification and this report. Most of the literature reviewed for D4.1 remains relevant for D4.2, and of course D4.1 itself constitutes the basis of the work done here.

The best practice report in D4.1 describes currently existing partnerships between the cultural heritage sector and commercial partners, as well as the media industries' best practice in terms of:

- Standard identifiers;
- Standard descriptive metadata schemas;
- Underlying conceptual models.

To summarise its key findings:

- Extremely rich metadata is available across sectors;
  - Marketing collateral means supplemental content is also available;
  - Standards are more or less mature, well-documented and interoperable, depending on sector.
- Conceptual models exist in the commercial and cultural heritage sectors;
  - Both of the main models are event-based and therefore basically compatible;
  - Semantic mapping across sectors & schemas is possible;
  - Some work must be done to overcome the difference in focus (see D4.1, section 4.3. and 5.4.).
- Commercial metadata has unique characteristics:
  - It consists not of repositories or catalogues, but of data flows between partners, to enable trading through the supply chain;
  - It must therefore be updated for changes in products, prices, availability, links and marketing collateral.
- It has an intrinsic legal–commercial aspect:
  - It is closely controlled and therefore relies on robust, independently administered identifiers to ensure provenance;

[23] Journal of Information Science (JIS); Journal of Librarianship and Information Science (JOLIS); Health Informatics Journal; IFLA Journal.
[24] "Data exchange" would have been another search term of relevance to this deliverable; however, there is a significant difference in this context between "integration" and "exchange", as explained in detail in, *e.g.,*Stein, *et al*. (2005).

- It is a significant commercial asset and qualifies for database right in the EU;
- It may itself contain extracts or derivations from creative works that are thus covered by copyright;
- It is often licensed for re-use.

- Therefore to use it in partnerships (such as the real examples in the report, and any future proposal) we need to develop:
  - An attractive and realistic business case;
  - A robust data licensing model.

All of these findings – especially those pertaining to conceptual models and commercial sector-specific requirements – will be referred to throughout this report at the appropriate point in the detailed discussion of the sector-specific mappings. Conveniently, all of the main insights apply clearly to the ONIX mapping which forms the central exposition of this report, but their relevance to the other schemas will also be noted where possible.

4.3 PREVIOUS AND CURRENT EUROPEANA PROJECTS

During the preparation of this report, Work Group 4 kept a watching brief on other projects in the Europeana network and beyond, both current and past.

Project	Domains addressed	Standards used	Tools created	Insights
ATHENA	Museum	LIDO	MINT	Metadata normalisation and harvesting pipeline described above (section 3.1.)
Europeana Libraries	Books	EDM	European Library Aggregation Architecture	EDM is not yet suitable for aggregating library data[25]
Europeana Photography	Photo	LIDO, IPTC	N/A [still in progress]	N/A [still in progress]
Europeana Connect	Music	DC (local application profile)	DISMARC	Need for on-going institutional commitment / investment to maintain / develop aggregators
EUScreen	Film and TV (AV)	EBU Core (local application profile of DC)	EUScreen portal	
European Film Gateway	AV	EN 15907	EFG portal	Generated new cataloguing rules to cope with lack of existing standardisation

[25] See *Report on the alignment of library metadata with the European Data Model (EDM) (D5.1)*, available at http://www.europeana-libraries.eu/web/guest/outcomes

Project	Domains addressed	Standards used	Tools created	Insights
ARROW Plus	Books, photo	ONIX-RS	ARROW infrastructure	Identifiers and standard descriptive format for images are desirable
LOD2	Books	*	LOD2 stack [many tools]	N/A [still in progress]
ResearchSpace	Museum	CIDOC-CRM	ResearchSpace data curation environment	N/A [still in progress]
Digitising Contemporary Art	Photo, AV	LIDO	N/A [still in progress]	LIDO is suitable for description of AV and image resources
HOPE	Archive, library, photo, AV	LIDO	N/A [still in progress]	LIDO is suitable for description of image resources

The overall impression so far from these related projects is that interoperability of complex creative media works requires a rich and flexible harvesting schema like LIDO, although this is not always realised, for example for music or AV recordings, nor for the complex information objects described by library metadata, which are not currently adequately described even by the updated Europeana schema.

The technical and semantic bases for creating linked cultural data are being put in place. It should be noted that although projects like LOD2 are investigating use cases and technical solutions for the commercial sector to publish linked open data, so far this does not seem to include commercial product or media asset metadata.

Importantly for Linked Heritage, the DCA, Europhoto and Europeana Photography are using LIDO to aggregate data for media objects very similar those considered here, and in some cases may use some of the same source data formats (*e.g.* IPTC/XMP). The presence of many library partners on the Linked Heritage project indicates that LIDO's suitability for aggregating book data from the heritage sector will soon be clearly understood, complementing the work done here on ONIX.

4.4 CROSS-DOMAIN MAPPING: BEST PRACTICE

Just as considerable expertise and best practice has been accumulated in creating and transmitting product data in the commercial sector, so, partly in direct consequence of this, a body of best practice in interoperability is also available. As was seen in D4.1, both the commercial sector and cultural heritage world have similar approaches to the problem, albeit with a different emphasis.

First we should clarify that the type of data integration described here is not federated search or federated query construction, such as is available *e.g.* through The European Library for simultaneous search access to the catalogues of the national libraries of Europe. Rather, it is integration of the contents of databases themselves; "data integration" proper[26]. The resulting integrated datasets should then be available for further reuse, such as aggregation into portals like Europeana.

[26] Sometimes referred to as "data warehousing".

4.4.1 The need for mappings: semantic and syntactic interoperability

Datasets from two different domains are "semantically interoperable" when the definitions[27] ("semantics") of the terms[28] used to create, select and combine the information they express are the same, or at least understandable in the same terms ("interoperable"). This process of understanding and using the information in the data can be more or less automated, but at some point must involve a "meeting of minds"[29] as this is the origin and purpose of all terminology (and indeed all language). The underlying purpose is to communicate record and use the concepts (and facts) represented by the information; without clear definitions of these concepts and their relations to the terms used, the data are meaningless and thus useless for communication.

This is the basic requirement for interoperable data; in practical use we can also require "syntactic" compatibility, the way that terms are combined to create usable information from data. The Dublin Core Metadata Initiative (DCMI) model of "Interoperability Levels for Dublin Core Metadata"[30] sets out a useful analysis of the levels at which this can be achieved using modern tools such as XML (described below) and RDF[31] (discussed later in the findings of this report – see section 14.1.3). These frameworks are designed to assist semantic and syntactic interoperability in the networked computing environment:

> "With networked information access to heterogeneous data sources, the problem of terminology provision and interoperability of controlled vocabulary schemes such as thesauri becomes increasingly urgent. Solutions are needed to improve the performance of full-text retrieval systems and to guide the design of controlled terminology schemes for use in structured data, including metadata."[32]

The fundamental needs underlying the semantic interoperability efforts of both commercial and cultural sectors are:

- Identification (of entities: physical objects and immaterial concepts);
- Contextualisation (through attributes and properties: description of objects and relations between objects);
- Access (to objects and potentially the above information about them for its own value).

These requirements clearly follow a certain chronological order, but the first two, identification and description, exist primarily to safeguard the third, access and proper use of the objects of interest. It is not immediately obvious that also concepts, including those used to describe "primary objects of interest" need to be unambiguously defined and identified, but this is actually fundamental to the whole enterprise, and this becomes obvious when parts of the process are automated:

"1. Obvious: Assign ID to resource.

- Once assigned, the number must identify the same resource;
- Beyond the lifetime of the resource, or the assigner.

2. Less obvious: Assign Resource to ID.

- The resource must be "identified";
- Must ensure it is always the same thing (bound);
- Describe the resource "content" [with precision];

[27] "Definition makes explicit… the… meaning of a term… A definition is symbolized by a general description, not by one word. A definition is a perfect general description." Joseph, M. (2002).
[28] "a term is a word, or symbol, conveying a particular meaning… to refer to a reality… [or] to refer to itself as a term or a concept, that by which we know, not what we know." Joseph, M. (2002).
[29] "When there is ambiguity in the communication of knowledge, all that is common are the words… For the communication to be successful, therefore, it is necessary for the two parties to use the same words *with the same meanings* – in short, to come to terms… *Every field of knowledge has its own technical vocabulary."* Adler, M. (1972).
[30] Nilsson, M., Baker, T. and Johnston, P. (2009). Available at http://dublincore.org/documents/interoperability-levels/
[31] See the W3C's official RDF primer at http://www.w3.org/TR/rdf-primer/
[32] Doerr, M. (2001). Journal of Digital Information, Vol 1, No 8. Available at http://journals.tdl.org/jodi/article/viewArticle/31/32

- Failure to do this will ultimately break interoperability.[33]"

The first point is the standard approach to "persistent identifiers" described in many European heritage and ICT project reports[34]. The second point is the recognition that (basic) shared terms and definitions for recognising and describing entities must be kept consistent and constant through time to avoid confusion when identifiers are used, and eventual loss of the relationship between the identifier and the entity it denotes (the referent) as language changes[35].

Identification of objects and concepts is primarily done directly by humans, at least initially ("even though we are convinced that the future lies in the coordinated combination of intellectual and statistical methods", as per Doerr, 2001). Description and relation of identified concepts can be done semi-automated fashion, but effective use of schemas (analogous to what DCMI call "Description Set [Profile]" interoperability; see footnote 22) is essential to this enterprise since every "item of metadata is a relationship that someone claims to exist between two entities"[36] and the forms these relationships can take must be identified and defined by schemas of some kind, as described in the next section.

4.4.2 Efficient, structured communication: controlled value lists, schemas and ontologies

It is helpful to more closely characterise the three main types of structured communication relevant to data mapping before moving on to the tools used in manipulating them. As can be clearly seen in the case of such structures as data dictionaries[37] the distinctions are somewhat fluid, with some data structures having members with the characteristics of others. Crucially, the kind of terms used distinguishes metadata schemas from the other two types:

"In considering whether a term is general or empirical, ask whether the term refers to the entire category of beings (general) or to an individual or individuals within that category (empirical)" (Joseph, 2002).

Applied to existing data models, the categorisation is as follows.

Data structure	Terms included	Characteristics	Prospects for mapping
Controlled value lists ("authority lists")	General or empirical (*e.g.* general terms for types of artwork for identification; empirical lists of artists' names for attribution)	Enumeration of terms, defined either implicitly by inclusion in the list, or explicitly in separate scope notes for each terms; provide commonly used data for information created according to a schema	Depend very heavily on the context of the whole list, level of definition for individual terms
Schemas	General only ("slots" for creation and aggregation of data about objects of interest)	Used to define general or specific values to be communicated about a defined class of individual objects, for a specific use context	Almost always possible from a more specific to a more general schema; loss of specificity occurs unless terms have identical meanings in both schemas

[33] Paskin, N. (2004). Keynote: The development of persistent identifiers. *ERPANET Persistent Identifiers seminar.*
[34] Including MINERVA, ATHENA and Linked Heritage.
[35] "Recognition" and "interoperability" are discussed in the DOI context at: http://www.doi.org/doi_handbook/4_Data_Model.html#4.3.1
[36] Rust, G. and Bide, M. (2000). *The indecs metadata framework.* Available at http://www.doi.org/topics/indecs/indecs_framework_2000.pdf
[37] For example, the indecs data dictionary found at the link in the above footnote, or the DDex data dictionary, linked from the footnotes of section 4.5.2.

Data structure	Terms included	Characteristics	Prospects for mapping
Ontologies	General or empirical (general terms used to harmonise schema models; empirical references used to create basic class definitions)	Highly specific definitions for all terms; relations between terms defined at least by hierarchical inheritance of class characteristics; provide underlying model for creating schemas and aligning definitions between them	As for schemas above; probably almost always possible using "linguistic" ontologies such as COA (see section 4.4.6) but not always practical

It is important to note that relative to other levels of description, each of these data structures can be considered an "instance" of a more abstract level. Thus data records created according to a schema are often referred to as "instance data" using the schema, even if what they describe is actually a general category (*e.g.* a class of products). A schema designed using a more general ontology can be considered in some sense an "instance" of the ontology's model; it is only an "instance" in a relative sense, since it defines a class of (unique) data records.

All three types of data structures can be modelled using XML, as described below.

4.4.3 Tools for mapping of XML data: schemas and XSLT

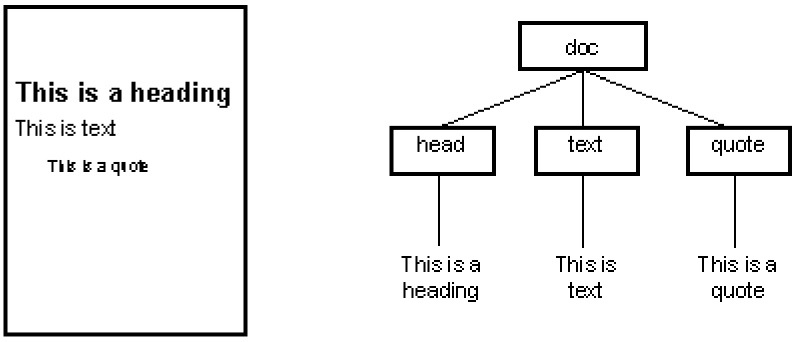
Previously, in D4.1, section 5.3.1, we contrasted the commonly-cited definition of metadata as "data about data" with the more fundamental indecs definition quoted above. In order to specify the problem of "schema mapping" it is helpful to note some characteristics of the data as it is found in practice, defining it in contrast to natural language:

- Metadata is highly specific and categorical. Because it acts as a surrogate for another resource or referent entity (usually an "information object" but certainly also a "heritage object") it needs to efficiently convey essential facts. Hence even less strictly modelled and defined data schemas use language far more concisely and "densely" than natural language. It tends to use categorisations to aid rapid description and clear identification.
- Metadata is highly structured information. The concept of a schematised method of entering, storing and retrieving data is used in many other fields, and, as there, in metadata management, its function is to promote standardisation in all three operations, to reduce the time and effort needed to make use of the schema and its related data.

Despite these strong contrasts in real practice, in theory "metadata" are still a kind of language, and the normal rules of language still apply. The study and practice of managing metadata is technical because grammatical and logical rules are applied strictly, for the above reasons, and to enable machine processing; they result in unusual and sometimes complex structures and frameworks such as those described here.

The XML (eXtensible Markup Language) standard is used to define the structure and other specifications of (mainly textual) documents in a wide range of fields[38]. Its primary characteristics are depicted in the following extremely simplified diagrams[39]. The first illustrates the use of XML to encode the structure of a simple text document. A unified document is broken down into nested or sequential "nodes" delimited by terms in angle-brackets. The essential features of "well-formed" XML are apparent here: there is one "root node" – in this case, the <doc /> node – which opens and closes the text serialisation, and all nodes have <opening> and </closing> instances of their name tags. Thus one could say that XML is a highly generic "structure" that can be specified for each use to provide "the" structural elements common to a set of similar documents.

[38] For an introduction to the formal W3C specification, see http://www.w3.org/standards/xml/core and for a technical introduction from another viewpoint, see http://www.xml.com/pub/a/98/10/guide0.html
[39] Taken from the UKOLN NOF Technical Advisory Service paper, *Metadata Sharing and XML* http://www.ukoln.ac.uk/nof/support/help/papers/metaxml/

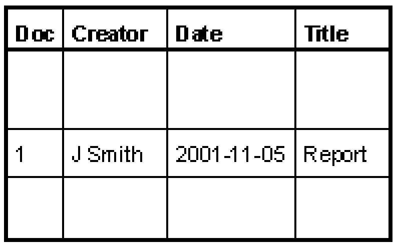


```
<doc>
<head>This is a heading</head>
<text>This is text</text>
<quote>This is a quote</quote>
</doc>
```

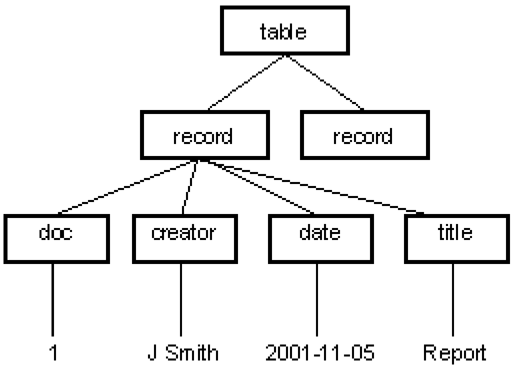
XML markup adding structure and semantics to a simple "document"

The XML markup itself carries only the document's *structure* and *content*. It is clear that some sort of presentation encoding[40] would also be necessary for the document's *appearance* to be reconstructed by the recipient of the XML serialisation. A content standard, defining what may, should and must be contained in a <doc> would be useful but not obviously essential. In the case of databases, we are dealing with a kind of information that is already to some degree constrained in its encoding and (more or less explicitly) typed, as in the second diagram (note the error in the "creator" element, however):

[40] Hence the common use of elements or attributes such as those mentioned in 7.2 and 7.3.3, indicating the intended use or meaning of information to display rather than store; specifics of presentation must be conveyed by formatting markup, for example HTML (http://www.w3.org/TR/html5/) and CSS (http://www.w3.org/TR/2011/REC-CSS2-20110607/).



Doc	Creator	Date	Title
1	J Smith	2001-11-05	Report



```
<table>
<record>
<doc>1</doc>
<creator>J Smith</text>
<date>2001-11-05</date>
<title>Report</title>
</record>
</table>
```

XML markup used to represent already existing structure and semantics of a database record

It is important to note that the XML example above consists only of data elements grouped under the root element <table> and the containing element <record>. Other containing elements could be introduced to add further structure and thus meaningful distinctions to data elements, such as <creator>, which could (for example) be split into <creatorName> and <creatorRole>, with the <creatorName> further separated into surname, first name, forms of address and so forth. In principle the level of elaboration of this kind, by elements, is unlimited since elements can always be added within the existing XML structure. In contrast, XML attributes are not present in the above example. These behave like data elements in that they contain text values, but not other attributes or elements. Thus they represent the limit of XML's extensibility and so are often used for data that applies very generally either to the document structure itself or to the raw data values. For example, the <date> element above might commonly be given a @dateFormat attribute to specify the encoding (in the example above the format appears to be YYYY-MM-DD). Note that an attribute might be used where an enclosed element would work just as well, for example, above the <creator> element might contain a @sortOrder attribute to distinguish first, second and third authors *etc.*, or equally <creator> might contain a sub-element <creatorProminence> or simply <sortOrder>. In such cases there is an element of judgement in the overall document design; however, attributes are most naturally used for the most generalisable variables or raw data encoding. Note that this is a key distinction between XML formats such as ONIX for Books, and other serialisations such as the MaRC family of formats; the MaRC fields for book measurements, page counts, *etc.* contain text which can include numbers but also letters denoting the units; in ONIX these are separated so they can be parsed more easily. This is not essential to XML but certainly its structure lends itself more naturally to the separation and specification of different types of encoding and semantic content.

Here, the need for an explicit schema to define the types of data, their relations to other parts of the database and their cardinality[41] will be extremely important if not essential to the data recipient trying to reconstruct the database and potentially merge these data with others. If the schema of any other data to be related to these differs significantly, a formal definition of the relationship between the schemas needs to be elaborated. This is the role, at least on a syntactical and algorithmic level, of XSLT[42], as it specifies how to transform data

[41] See D4.1, section 6.1. for an explanation of "cardinality".
[42] See the W3C pages on XSLT at http://www.w3.org/standards/xml/transformation

conforming to one XML schema into data that conforms to a different XML schema[43]. Software solutions for implementing XSLT and similar mapping languages will be discussed below in section 4.5., but note here the essential point that a data element ("node") in an XML tree can be reference through an XPath expression which denotes its "path" from the root node. To take the example diagram above, the XPATH reference for <date> elements can be expressed as "table/record/date". To specify the <date> of the first <record> serialised, a function is needed, *e.g.* "table/record[position()=1]/date" (i.e. note that both table and data are not repeatable). This is the kind of operation[44] on the syntax of XML representations of data that are used in syntactic mappings. The relationship between syntax and semantics will be further explored below in sections 4.4.4. and 4.4.5.

Although relations between the elements' meanings (terms) can be defined through the logical structures of XML (and perhaps RDF), at some point, the values of the elements' content must be defined in a natural language comprehensible to its creator and user. This level of definition is not currently automated and there must be at this point a "meeting of minds" at the level of human interaction. As Godby (2012) recently noted after completing the updated mapping of ONIX for Books to MaRC21: "the barriers to progress [in interoperability] are cultural or political, not technical". Hence we will look to a combination of technical and collaborative mapping solutions, outlined in the next three sections.

4.4.4 Complementary approaches for schema mappings

Godby, Smith, and Childress (2003) outlined two familiar approaches to semantic mappings of metadata schemas originating in different domains:

- the short translation path – a pre-defined transformation is applied to the schema, automatically mapping each source element to a target element;
- the long translation path – each source schema is mapped by hand to a core ontology, which then maps to the target schema.

The first approach is the so-called "crosswalk", of which many examples already exist. Although not necessarily the case, crosswalks have tended to be associated with relatively ad-hoc harmonisations between schemas, not necessarily authoritative, and different versions may give quite different results. The longer route, sometimes called "hub-and-spoke", depends on a robust core (or "mediation") ontology capable of expressing terms in potentially very different knowledge domains. Each of these has its advantages and problems:

Mapping approach	Pro	Con
Short	Potentially quick checking of correspondence by human experts on either side where terms and syntax are relatively simple to understand and well-documented	Can tend towards ad hoc mappings where correspondence is at best partial or ambiguous; mapping decisions not always well documented; no suggestions for new, common terms to accommodate areas not yet shared by both formats
Long	Only one mapping per input schema is required so long-term efficiency results; detailed analysis can pinpoint existence and also degree of correspondence between terms	Detailed ontological analysis far more time-consuming; can suggest improvements and additions to either or both formats

The two approaches to schema mapping outlined above form a spectrum that covers most activity in this area. They are not antithetical, as a direct crosswalk could indeed be generated from a hub-and-spoke mapping analysis.

[43] XSLT is certainly not the only language that can be used for this purpose – for example, the related Xquery is also used, and OCLC (Godby, Smith, and Childress, 2003) even developed a proprietary language achieving the same results – but XSLT is the language implemented so far by Linked Heritage.
[44] For more on XSLT and XPATH functions, various reference documents are available, for example from MicroSoft http://msdn.microsoft.com/en-us/library/ms256069.aspx

A third approach, that of statistically inferring mappings (see Doerr, 2001) from actual data produced using each schema, for example by frequency of data values and their co-occurrence, would certainly be interesting, and potentially useful to both types of approach outlined above; but it falls outside the expertise of the Work Group and would probably be beyond the resources of the Linked Heritage project (and of Europeana).

In order to mitigate the difficulties experienced in creating automated or user-defined mappings, the focus of most practical work turned to improving the semantic detail in the core ontology[45]. This has two aspects, the event-based data modelling approach found in both the content industries and the cultural heritage sector (see Linked Heritage, D4.1; section 5.4), and the architecture of the ontology itself. These will be briefly outlined in the next two subsections.

4.4.5 Conceptual Models (CIDOC-CRM, FRBR(oo) & Indecs)

As noted in D4.1, the two main communities of practice in question here have produced core ontologies based on their domain knowledge and practice. These share the two main characteristics necessary to their function: a distinction between conceptual and perceived objects of interest, and an analysis based on the “event” (or “context” – the terms are used interchangeably at least in the commercial data world). The FRBRoo model is of particular interest here as it anticipates the work of Linked Heritage by incorporating the commercial world’s object of interest (for the book sector) into the heritage world’s focus for description (contextualisation):

Model	Object of primary interest	Contextualisation
Indecs	Commercial products	Product life cycle
CIDOC-CRM	Cultural heritage objects	Object’s “life history”
FRBRoo	Books (generally commercial products but in theory some could be unique heritage objects)	Product lifecycle as if it were a “life history”

The difference in focus of description is best seen in the terms used to describe the event types used to generate other classes and properties. Some examples taken from the immediate sub-classes of the “event” class in each model are shown below (note that terms like “expression” are defined differently in indecs and CIDOC-CRM/FRBRoo):

[45] At roughly the same time, the 1990s, in the heritage, library and copyright content sectors. See D4.1 and http://www.doi.org/topics/RustModelofMaking2005.pdf for a discussion of their similarities.

CIDOC-CRM[46] **(life history events)**	**Indecs[47]** **(product lifecycle events)**	**FRBRoo[48]** **(product lifecycle events as life history)**
Activity	assertion	Activity
- Modification	creatingEvent	- Performance
- - Production	disseminatingEvent	- Creation
- - Part Addition	expression	- - Work Conception
- - Part Removal	transaction	- - Expression Creation
- Attribute Assignment	transformingEvent	- - - Recording Event
- Creation	usingEvent	- - - Publication Event
Beginning of Existence		Beginning of Existence
- Birth		- Production
- Transformation		- - Expression Creation
- Production		- - Carrier Production Event
- Creation		- - Reproduction Event
End of Existence		[further as in CIDOC-CRM]
- Destruction		
- Death		

Here it can be seen that through the FRBRoo analysis of authorship and publication of books, the event types associated with mainly conceptual creations (intellectual property) typical of indecs have started to find a home within the more general historical apparatus of CIDOC-CRM. It must be noted, though, that the FRBRoo framework is based in some key areas on CIDOC's draft "MetaCRM"[49] which allows (among other things) detailed descriptions of class properties. Neither FRBRoo nor MetaCRM are yet officially incorporated into CIDOC-CRM, or yet implemented in LIDO. Therefore at this stage FRBRoo can only be used to inform the mapping work, but not fully relied on for future interoperability.

The indecs ontology informs ONIX for Books and DDex. This can be seen very clearly in the fact that distinct messages from the ONIX family of standards describe two of the main entities in the indecs model[50]:

- indecs:abstraction – ONIX for ISTC (registration message for abstract texts);
- indecs:manifestation – ONIX for Books (and ONIX for ISBN, a subset thereof).

The ISTC manual also makes clear that the abstractions identified by the ISTC are defined in terms of events (origination or derivation) with identifiable actors involved. These definitions and distinctions are in harmony with the indecs "model of making" and more general event basis.

Another more subtle example is seen in the enhancements to the ONIX for Books standard, from version 2.1 to the current version, 3.0. For many concrete examples, see Godby (2012), on which the illustration below is based. In the most recent ONIX schema, the semantics of many elements that previously had extremely specific term definitions are incorporated into composites that spell out the same information step-by-step, for example, some of the various subject classification elements:

[46] Taken from Crofts, N. et al. (2011). *Definition of the CIDOC Conceptual Reference Model*. Available at http://www.cidoc-crm.org/docs/cidoc_crm_version_5.0.4.doc
[47] Taken from Rust, G. and Bide, M. (2000). [and see footnote 26].
[48] Taken from Bekiari, C., Doerr, M. and Le Bœuf, P. (2010). *FRBR object-oriented definition and mapping to FRBRer (Version 1.0.2)* Available at http://www.cidoc-crm.org/docs/frbr_oo/frbr_docs/FRBRoo_V1.0.2.pdf
[49] For discussion of CIDOC MetaCRM and other related drafts, see http://www.cidoc-crm.org/working_editions_cidoc.html
[50] See the discussion of the indecs data model in D4.1 section 5.4.3.

ONIX version	XML encoding	Natural language meaning(s)
2.0	<BICMainSubject>GM</BICMainSubject>	"The main subject code for this book, in the BIC subject classification scheme, is GM"
3.0	<Subject>	"A subject of this book…
	<MainSubject/>	…specifically, the main one…
	<SubjectSchemeIdentifier>12</SubjectSchemeIdentifier>	…as encoded in BIC subject classification…
	<SubjectCode>GM</SubjectCode>	…is GM
	</Subject>	"

In the ONIX version 3.0 example, the BIC subject scheme is identified by the code "12" from a code list stipulated by the ONIX specification. Such flexibility was partly available in ONIX 2.1 but there were dedicated XML elements for subject codes taken from particular schemes (BIC, BISAC), and for 'main' subject codes. In analysing down the element terms more finely, the ONIX 3.0 schema actually reduced the total number of terms needed.

The approach whereby each type of data is made as general as possible is representative of the overall ONIX design, which also allows "proprietary" as a type for many data items, notably identifiers. This contrasts with the approach of LIDO, which strongly favours published identifiers (see section 8.2), whereas ONIX follows the indecs principle of retaining provenance information as essential data.

Note finally that because of the added generality and reuse potential of the data and structural elements, still further implicit relationships could potentially be analysed out of the ONIX 3.0 element terms in a new version of the schema;, for example, to answer the question, according to whom is the subject GM the "main" subject of the book? Classifications assigned by various different agencies, perhaps the publisher, booksellers, and libraries, might each consider the main topic to be something (probably only slightly!) different; thus an extra element within the <Subject> composite, perhaps designated <SubjectAssignmentAgency> and needing various sub-elements to unambiguously identify the agency, might be added (if the need for this detail were proposed by libraries and historians of publishing, perhaps).

However, in defining the XML schema and its underlying semantics, EDItEUR has in effect codified which types of relationships the ONIX standard's users will need. Any other standard schema, including LIDO, will draw the boundaries elsewhere, and thus some relationships may not be expressible in both (almost a certainty unless the target schema's terms are so general as to be practically unusable).

Any mapping thus needs to take into account not only the semantic definitions and syntax expressing them in the source and target formats, but as developed in the discussion above, the best practice and other contexts of the use case for the formats involved, and the mapping itself. This leads us to consider the final, most abstract ramification of schema mappings in the next section.

4.4.6 Contextual ontologies and the Vocabulary Mapping Framework (VMF)

The work of the indecs project in 2000 was followed up by a more generalised metadata modelling framework project (CONTECS, in 2001[51]). This in some senses applied the methodologies of indecs to the process of assigning metadata itself and resulted in a highly generalised schema (OntologyX, now managed by RightsCom[52]) that can be used to perform the kind of analysis of relationships and meanings described above at the lowest possible level of logical granularity. The analysis is based on an event structure similar to CIDOC-CRM, with the central key concept of the "context" defined by the kind of activity or change taking place, thus defining the semantics of "verbs" such as "create", "publish", "produce", or "acquire".

[51] See Paskin, N. (2004).
[52] See the RightsCom homepage for OntologyX at http://www.rightscom.com/Default.aspx?tabid=1067

In the diagram below it is presented as the "richest, most indirect" view of a given set of metadata, of the three possible levels common to schemas (attribute level), graph representations like RDF (relationship) and contextual ontologies. The contextual, or event analysis can be used to model any type of data in a maximally generalised and interoperable way.

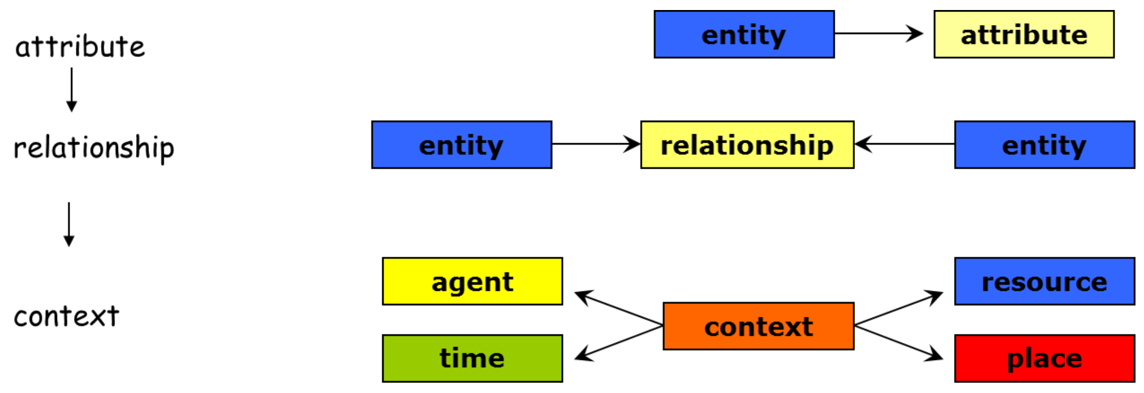


Figure 1 – "Ontology approach: deeper view of metadata" from Paskin, N. (2004)

In this way the OntologyX schema (the "Contextual Ontology Architecture", or COA schema), however, limits itself to expressing the linguistic and logical meaning of data in the context determined by its input terms, leaving ontological definitions in the knowledge domain of interest to the relevant experts. Therefore it can be used to represent not just the data but the schemas and ontologies themselves for purposes of integrating their data and creating new messages across knowledge domains where needed, if these do not yet exist.

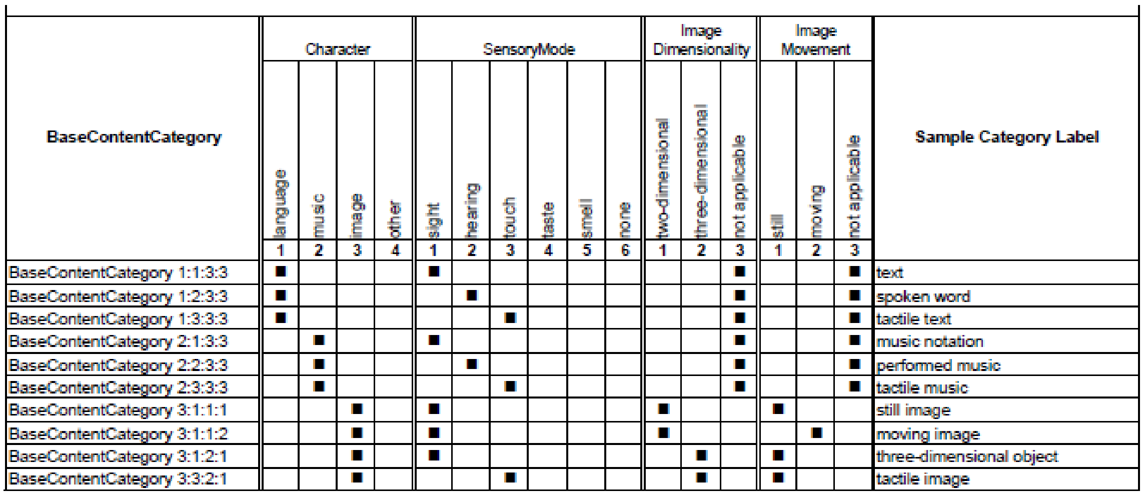
This work was of great value to the commercial content sector because of the necessity to create precise, reliable machine-processable expressions of rights and use policies for intellectual property content (for example, through the MPEG Rights Data Dictionary[53]), but the same "toolkit" of conceptual (really linguistic and logical) analyses and data management structures was also seen to have potential value for library and other heritage sectors. The JISC-funded VMF project in the UK (running June-November, 2009; homepage: http://www.doi.org/VMF/) applied the OntologyX schema to parts of schemas and at least one existing authoritative mapping[54] of a small set of terms with a narrow range of meanings, primarily about the format and medium of creative media manifestations, across commercial and heritage sector schemas and vocabularies. The schemas included were:

- CIDOC-CRM
- DCMI
- DDex
- FRAD
- FRBR
- IDF
- LOM (IEEE)
- MaRC21
- MPEG21 RDD
- ONIX for Books
- RDA
- RDA/ONIX Framework

[53] See the RightsCom page on "Rights Data Dictionary (RDD)" at http://www.rightscom.com/Default.aspx?tabid=1172
[54] Such as the entire RDA/ONIX framework, found at http://www.rda-jsc.org/docs/5chair10.pdf

It is clear that if LIDO, and the relatively self-contained and well documented schemas for IPTC Core and Extension, and the EIDR referent metadata, were included, and the remainder of the ONIX for Books and DDex schema elements mapped, an incredibly rich mapping resource would result, covering the whole scope of Linked Heritage D4.2 from the semantic perspective, and opening the door to producing a new, comprehensive harvesting schema for any commercial product information or cultural heritage object. However, this would involve very significant work and expense (see section 4.5.3) that is certainly beyond the scope of the current project and may exceed the value of the benefits.

A formal concept analysis[55] of each term produced "atomic" categories such as those below taken from the RDA/ONIX Framework.



BaseContentCategory	Character				SensoryMode						Image Dimensionality			Image Movement			Sample Category Label
	language	music	image	other	sight	hearing	touch	taste	smell	none	two-dimensional	three-dimensional	not applicable	still	moving	not applicable	
	1	2	3	4	1	2	3	4	5	6	1	2	3	1	2	3	
BaseContentCategory 1:1:3:3	■				■								■			■	text
BaseContentCategory 1:2:3:3	■					■							■			■	spoken word
BaseContentCategory 1:3:3:3	■						■						■			■	tactile text
BaseContentCategory 2:1:3:3		■			■								■			■	music notation
BaseContentCategory 2:2:3:3		■				■							■			■	performed music
BaseContentCategory 2:3:3:3		■					■						■			■	tactile music
BaseContentCategory 3:1:1:1			■		■						■			■			still image
BaseContentCategory 3:1:1:2			■		■						■				■		moving image
BaseContentCategory 3:1:2:1			■		■							■		■			three-dimensional object
BaseContentCategory 3:3:2:1			■				■					■		■			tactile image

Figure 2 - "Examples of base content categories" from AACR JSC. (2006).

For example, to say a resource consists of "text" would mean it is made up of "language" (only) for interpretation by "sight" (only), without any aspect of "dimensionality" (2D or 3D) or "movement" (still or moving). The COA analysis was then applied to assign verbs defining the "context" *e.g.* of creating, adapting or translating a "text" so defined, and the resultant "resources" and "agents" assigned places within the VMF ontology as in the example below:

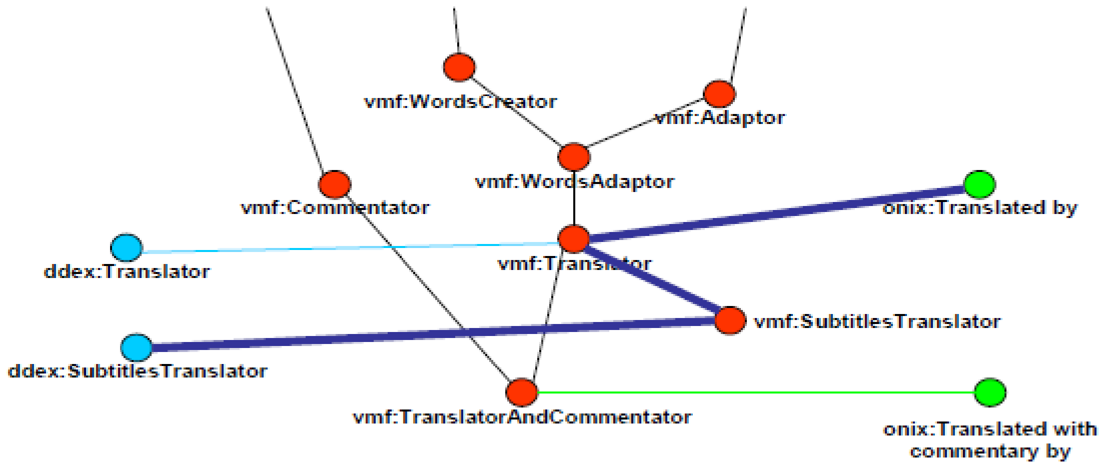


Figure 3 - "Structure of the VMF matrix" from Rust, G. (2009).

In the above figure, one can see terms from the input schemas (ONIX and DDex) in blue and green, mapped exactly to VFM terms, in the larger VFM hierarchy. The bold lines indicate "best fit" mappings between input

[55] See Sowa, J. F. (2007). *Conceptual Structures: Mathematical Background.* Available at http://www.jfsowa.com/logic/math.htm#FCA

schemas via the VMF ontology. Such programmatically generated schema mappings were among the proposed use cases for VMF, which included:

- Searching/querying across multiple data formats;
- *Taxonomy/subject index mapping;*
- Mapping of local, bespoke metadata schemas;
- Preservation metadata;
- *Full message transformation (metadata crosswalk)*[56].

Obviously the two use cases in italics above have great relevance to Linked Heritage. Mapping taxonomies and subject terms falls under the remit of Work Package 3, and full message transformation (crosswalk) is the task of the content coordination or specification partners in WGs 4 and 6. Crosswalks produced using the VMF would have the additional advantage of being "short path" translations based on "long path" analysis (see section 4.4.4).

Notwithstanding its unfamiliar and often very abstract terminology, the VMF "language" could also be used to construct new, self-contained sets of elements for multimedia aggregation and discovery environments, in much the way that the Dublin Core elements were originally conceived. For the core element set, the most concrete shared terms whose definitions include by subsumption (*i.e.* "sameAs" or closest "superClassOf") all the mandatory elements of each schema would be used, perhaps with the addition of the "recommended" elements of each scheme as identified by their communities' best practice guidelines. The difference between this approach and *ab initio* selection of "universal" core elements is that the semantic links to source schemas would have already been articulated in detail, so there would be no need to create qualified terms, application profiles and local practices to make up for the deficiencies and ambiguities of the basic elements.

4.4.7 LIDO as an instance-level CIDOC-CRM implementation

At this point LIDO can be mentioned as an aggregation schema that does already implement some of the features of such an ontology-based core element set. Its terms are extremely general because they rely on the classes and relations of CIDOC-CRM; they are also deliberately selected for closeness to a range of instance-level domain schemas taken from across heritage collection management practice; the LIDO schema itself has some of the other distinctive features of the VMF matrix, in that alongside its role in harmonising its input data with the CIDOC-CRM, it also captures some relations to the input schema, through the @encodinganalogue attribute available for many elements. This ensures that the link to the original definition of the data is preserved even though LIDO's categories are usually more general. However, because it is currently conceived as mapped to a specific portion of CIDOC-CRM (for physical objects) its overall semantic range is limited. As will be explained below (section 5.1, with more details in Appendix 2) a more flexible mapping to allow corresponding LIDO properties to be expressed for conceptual objects (specifically classes such as product types) would bring LIDO into line with the FRBRoo extension and closer to the indecs/COA approach.

4.5 EXISTING MAPPING, AGGREGATION AND DISCOVERY SERVICES

This final stage of the literature review briefly highlights some of the project- or service-scale implementations of the above best practice, in the heritage and commercial sectors.

4.5.1 MINT

Linked Heritage subscribes to the MINT aggregation and mapping software platform hosted by NTUA[57]. For each project using MINT, an aggregation schema (for Linked Heritage, of course, this is LIDO) and a publication

[56] Compiled from the VMF home site at http://www.doi.org/VMF/archive.html and final report at http://www.jisc.ac.uk/media/documents/projects/vmf_final_report.pdf

[57] For an introductory overview of MINT see http://mint.image.ece.ntua.gr/redmine/projects/mint/wiki/Introduction and for technical details in relation to Linked Heritage see http://www.athenaeurope.org/index.php?en/149/athena-deliverables-and-documents (deliverables D7.1 and D7.4), and http://www.linkedheritage.org/index.php?en/142/documents-and-deliverables (deliverables D5.1 and D5.3).

schema (here, ESE) is specified[58]. Data can only be uploaded; MINT is not able to update metadata publications record-by-record, so in the case of Linked Heritage contributions to Europeana would have to be updated[59] in negotiation with Europeana's aggregation team. MINT ingests (among other formats) XML instances, empirically generates an input schema based on the XML instances input, and allows the user to map this schema to LIDO, generating an XSLT script for the transformation. Hence the experiments described in this report used XML instance data contributed by the relevant standards organisations, experts in the domain (IPTC) or licensed implementers of the standards (DDex). MINT currently uses a single, standardised mapping of LIDO to ESE[60].

4.5.2 Linked Heritage Terminology Management Platform (TMP)

Controlled value lists for enrichment of the Linked Heritage aggregation data will be managed with a bespoke TMP developed in part from xTree[61]. The format for aggregating terminologies is SKOS[62], a data modelling language for representing existing controlled vocabularies. Its structure has similarities with languages for expressing formal ontologies, but is meant primarily for lightweight representation and retrieval, rather than extensive modelling of complex relationships[63] (for example, it does not define any relationships between concepts beyond hierarchical links and generic "relations"). The vocabularies to be expressed in SKOS, in as far as they are hierarchical and consist of a concept code (SKOS:notation), label and scope note (or definition), have much in common with the controlled value lists used in commercial metadata, for example:

Commercial metadata standard	Terminologies (mostly excluding schema elements)
ONIX for Books	ONIX code lists[64]
DDex	DDex data dictionary[65] (parts)
EIDR	EIDR schema(s) enumerated values[66]
IPTC / XMP	IPTC newscodes[67]

Further, at least one of the standards, ONIX for Books, contains elements that can hold values from other vocabularies (especially subject codes) widely used in the commercial and heritage sectors, and of course it can contain values from proprietary vocabularies too.

As the LH TMP is currently still in development, and MINT, as discussed below, is not yet integrated with it or ready to accept other SKOS imports, only important points of contact with controlled vocabularies will be dealt with in this report.

[58] Hence MINT can be used in the Linked Content Coalition successor project, RDI, to map a wide range of rights and licence expressions to a common model. See: http://www.linkedcontentcoalition.org/uploads/1206120_plenary.pdf
[59] See D4.1, sections 5.3 and 9.1 for the need for updates and provenance of metadata, as well as the findings and conclusions of this report.
[60] See Stein, R. *LIDO v1.0 to ESE v3.4 mapping table.* Available at http://www.linkedheritage.org/index.php?en/177/training-material-targeted-to-linked-heritage-content-providers#6
[61] See the W3C page for xTree at http://www.w3.org/2001/sw/wiki/XTree and ATHENA documentation at www.athenaeurope.org/getFile.php?id=583
[62] The official SKOS primer is available at http://www.w3.org/TR/skos-primer/ and Linked Heritage's guide for SKOS implementation can be downloaded at http://www.linkedheritage.org/getFile.php?id=244 (and see also Wyns, R. and Leroi, M. (2012). *D3.1 Best practice report – Terminology.* Available at http://www.linkedheritage.org/getFile.php?id=286)
[63] See the comparison of SKOS, RDF and OWL at: http://www.w3.org/TR/2009/REC-skos-reference-20090818/#L1045

[64] http://www.editeur.org/ONIX/book/codelists/current.html
[65] http://ddex.net/dd/dd_ERN35_DSR41_MWL21/
[66] http://www.eidr.org/schema/1.0/
[67] http://iptc.cms.apa.at/site/NewsCodes/View_NewsCodes/

4.5.3 OntologyX

The OntologyX architecture and VMF are still available for use but would require significant extra project time and possibly subcontracting[68] and so falls outside the practical scope of LH WP4. It would be highly desirable to build on this work in possible future projects or longer-term work, so possible ways forward will be discussed in sections 14 and 15. Methodologies for Producing Experimental Mappings

This comparison of methods draws on the literature reviewed above, and makes the link between solution of the problem in theory and the experimental work done for T4.4.

4.6 MAPPING PRODUCT CLASS DATA TO AN INSTANCE SCHEMA

The conceptual background to the mappings presented here has this basic problem at its heart: LIDO is a schema for unique, single items curated by heritage institutions; the source schemas for these mappings are abstract classes of commercial products (and in the case of EIDR, can be even more abstract classes of product classes). A product class is defined by the publisher or other "releasing" company but has no "repository" as such, and also lacks other key attributes which only items can possess.

A fuller discussion of solutions to this problem is found in Appendix 2. Here we will simply note some factors that tell in favour of adopting the simplest solution, that of mapping each class property to the relevant item property where available, and signalling this in the data record itself:

- The fundamental conceptual modelling work to enable this solution has already been done[69], and the benefits to the heritage and library sectors are well established. This approach will extend the existing scope to include *e.g.* audiovisual archives, music recording archives and photography libraries:

"Mediation tools and Semantic Web activities require an integrated, shared ontology for the information accumulated by both libraries and museums for all the collections that they hold, seen as a continuum from highly standardised products such as books, CDs, DVDs, etc., to raw materials such as plants or stones, through "in-between" objects such as draft manuscripts or engraving plates. In addition, such typical "library objects" as books can be *about* museum objects, and museum objects can represent events or characters found in books (e.g., 'Ophelia's death') and descriptions of museum objects in museum databases may contain references to bibliographic resources that mention those museum objects: such interrelationships should be… integrated in common information storage…"[70]

See also some initial proposals for modelling specifically commercial products such as art prints, replicas, CDs of archival sound, and of course books and DVDs, in Appendix 3 of this report, kindly contributed by Patrick le Boeuf, one of the primary authors of FRBRoo.

- Modelling of "types" as in FRBRoo is inherently useful for heritage work and could have unanticipated benefits by modelling, for example, conceptual classes within the content of intellectual objects (subject terms but also conceptual constructs like narratives, references, philosophical formulations):

"…types play a central role in the history of human understanding; they are intellectual products, and documentation about the history and justification by physical evidence of types… falls squarely within the intended scope of the CRM…"[71]

- The existing use of LIDO points towards extension to cover commercial publications and releases of all kinds; the main features and characteristics of these product classes are shared entirely with collections of ephemera, archives of broadcast and cinema media, sound archives, Web archiving initiatives, museums of publishing, technology, contemporary digital art, *etc.* In any case, LIDO can and is being used in libraries as part of Linked Heritage.

[68] See for example the standardised costs of mapping new terms to the VMF: http://www.doi.org/VMF/registering.html
[69] That is, in FRBRoo and meta-CRM: http://www.cidoc-crm.org/frbr_inro.html and http://www.cidoc-crm.org/working_editions_cidoc.html
[70] FRBRoo version 1.0.2, available at: http://www.cidoc-crm.org/frbr_drafts.html
[71] CIDOC meta-CRM draft, available at: http://www.cidoc-crm.org/working_editions_cidoc.html

Having identified the main forward step inherent in this cross-domain mapping, possible practical methodologies are now described, beginning as before with the most general and moving towards the most practical and concrete. It should be noted that each of these approaches involves doing the same substantial work: modelling data and use cases; analysing, identifying and matching semantic content of terms; selecting best fits across the schemes. The difference is only in the methodology, the conceptual frameworks and procedures, and technical tools used; all are valid approaches but each has pros and cons. This report appears timely as at the time of writing, a draft International Standard is under review, which synthesises general recommendations for creating schema mappings[72]. In all cases, the work will be time-consuming, requires skilled practitioners with significant domain knowledge, and is justified only where there is reasonable expectation of large-scale mapping of data from one domain to another. In the comparison tables below, decisive factors, either pro or con, have been highlighted to make it clear how the current approach was selected.

4.7 MAPPING BASED ON AN UPPER ONTOLOGY

In this approach, represented in the extreme case by the VMF (section 4.4.4. above), a top ontology is created from very granular analyses of each term's natural language meaning (its "primitive semantics"). The relations between these new, analytical terms are extrapolated using a pre-defined analysis of the complete set of possible entities and relationships (a fundamental data model), until they provide links between all terms that must be mapped.

Aspects	Pro	Con	Solutions
Semantics	Extremely rigorous	May be too abstract to produce actionable results	Refer to data sample(s) Compare with other ontologies, especially in the relevant domain Produce new schemas based on shared semantics and use case
Completeness	Can easily include all elements as desired	Possibility of wasting time mapping all possible terms?	Work to specific use case for each mapping
Practicality	Produces authoritative and reusable mappings	**Initial analysis can be time-consuming and resource intensive** Maybe difficult to document results Overall complexity of process, high skill requirements, requirement for extremely broad domain knowledge	Automation of some processes Outsourcing of some processes Reuse of primitive semantics and terms from other ontologies

4.8 DIRECT MAPPING OF ELEMENTS

In this approach, very common in "crosswalks" between diverse fields, the schemas for the source and target data are compared, using the standard definitions and examples given in the schema specifications. Any fields that share sufficient meaning to satisfy the mapping's use case are considered to map (at least in one direction) and suggestions may be made to extend the target field to include new semantics if needed.

[72] ISO/DIS 25964-2, Information and documentation — Thesauri and interoperability with other vocabularies — Part 2: Interoperability with other vocabularies

Aspects	Pro	Con	Solutions
Semantics	Based directly on standards without potentially misleading examples **Could clarify (mis)matches within use case(s)**	May be abstract	Refer to data sample(s)
Completeness	Can easily include all elements as desired	Possibility of wasting time mapping all possible terms?	Work to specific use case for each mapping
Practicality	Uses a minimum of data and tools	May be difficult to document results May be time-consuming for large schemas Requires significant knowledge of two or more domains and of two or more schemas	Use standard templates where available Collaborate with source and target schema authorities

4.9 MAPPING EXEMPLARY INSTANCES WITHIN AGGREGATOR

Tools like the ATHENA / Linked Heritage MINT aggregator incorporate schema mapping into the aggregation workflow. Individual elements are mapped “manually” to the target schema (in MINT, this is LIDO) from a sample of XML data. This sample may be drawn from real life use, or created as an illustrative “dummy record” or be a mixture of both.

Aspects	Pro	Con	Solutions
Semantics	Maybe clearer from context of example data	Danger of using idiosyncratic records in sample Relative simplicity may be misleading **Aggregator schema may be fixed**	Use schema specification to clear up ambiguities Check against other samples
Completeness	Should include most common or typically used elements	May not include all elements needed for larger datasets	Examine real datasets Create dummy records / messages including elements not in sample data
Practicality	Use existing tools to automate some processes	Time and effort to learn tools Requires significant knowledge of two or more domains and of two or more schemas	Use standard mappings where available Collaborate with source and target schema authorities

A further difficulty of the exemplary instance approach is that all the schemas considered here, apart from IPTC/XMP, can appear in a variety of configurations depending on the type of entities described; for example, ONIX may describe single volume book products, or composite products made up of multiple volumes, or books plus CDs; DDEX may describe a release made available as products in diverse media such as CDs and digital downloads; and EIDR assets can be of a large number of types, classified at several levels of abstraction. Thus, there may be no one complete mapping of a whole schema.

4.10 APPROACHES CHOSEN FOR THIS REPORT

Based on these considerations, the mapping methodology used in practice for this deliverable comprised the following steps incorporating aspects of all three approaches:

1. Generate sample data using all mandatory and commonly used elements of
    a. Source schema;
    b. Target schema.
2. Upload sample source data to MINT.
3. For each section of target schema in MINT
    a. Analyse semantics of target schema elements;
    b. Compare semantics of source elements keeping in mind
        i. Analysis method used in top ontology approach;
        ii. Specification of source schema and best practice notes;
        iii. Specification of target schema and best practice notes.
    c. Select appropriate source elements and map.
    d. Add source conditions based on specifications and best practice.
    e. Update sample data with elements not present in order to create mappings for all relevant elements of the source schema, and repeat the above steps 2 to 3.e.
    f. Document mappings:
        i. Successful elements;
        ii. Elements and rules not yet possible in LIDO and/or MINT. Outline possible enhancements to schema and mapping tool that would enable expression of source semantics.
    g. Submit XSLT and source / output data to experts in source and target semantics for initial review.
    h. Test completed mappings with a variety of test data.
4. Document final mappings.

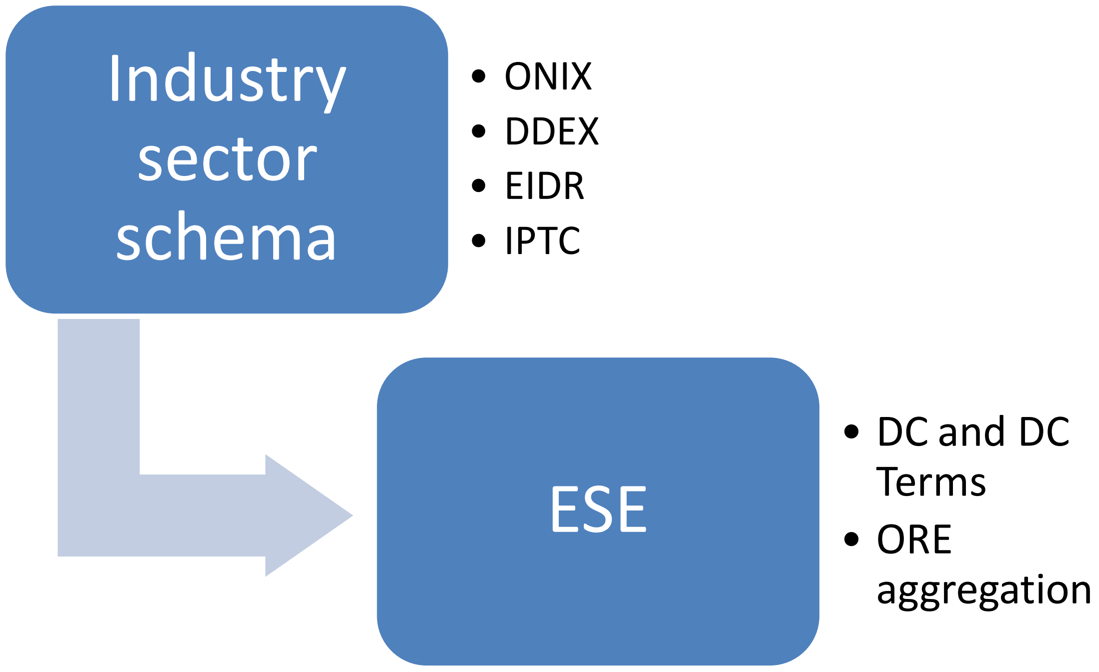
So far this methodology has only been used in full for the ONIX for Books 3.0.1 schema. For the other three areas work has been started and will be published when complete. Findings from each sector’s experimental mapping have been included here as indications of progress and the main problems to solve.

5 MAPPING COMPLEXITY FOR THIS PROJECT

The final question to answer before completing the practical work of T4.4 was to decide the appropriate level of detail for Linked Heritage mappings. Although at first glance it might appear reasonable to map all four sectors' schemas to LIDO, ESE and EDM, in practice this is a central problem for Linked Heritage and Europeana itself. As in D4.1, the decisions made in this methodological section represent part of the findings and are reflected in the recommendations and work plan towards D4.3.

5.1 MINIMAL MAPPING – DIRECT TO ESE

An initial hypothesis was to begin the mapping exercise from the broad context of Linked Heritage as a Europeana contributor and produce a mapping directly into ESE[73]. This would have the advantage of creating a potentially small and simple mapping (corresponding to the lightweight ESE schema) and allowing contributors from the commercial sector to submit data to Europeana without entering into partnership with Linked Heritage should they so wish.



Minimal mapping – industry sector schema to ESE

Experimental mappings of ONIX 3.0 and IPTC to ESE soon made apparent the impracticability of this approach, for general and sector-specific reasons, albeit with positive lessons learned, as summarised below:

Schema	Difficulties	Lessons learned
All (general aspects of ESE)	Both the small number of elements and the lack of appropriate semantic equivalents for many core properties of commercial data make a technically useful mapping impossible.	Any direct mapping would really be a selection of elements based on the individual use-case of specific providers, primarily chosen for display to customers.

[73] This has recently been done for previous extremely small-scale pilots of publisher data integration into Europeana. See for example: http://www.europeana.eu/portal/search.html?query=penguin&qf=PROVIDER:Penguin (13 books) or http://www.europeana.eu/portal/search.html?query=libreka (36 books)

Schema	Difficulties	Lessons learned
ONIX 3.0 and 2.1	The ONIX for Books schemas are so large, and offer so many options and combinations of elements, that any mapping to ESE will be extremely complex, without doing justice to the full semantics (especially for version 3.0.1).	A useful specification for the mapping to ESE must be based on an initial LIDO aggregation. The aggregator platform must be developed to allow either direct editing of the LIDO-ESE mapping (to produce data-provider-specific mappings), or to allow automated selection between a large number of optional elements based on complex business rules (beyond the ONIX schema).
IPTC	The IPTC properties, as expressed in XMP, at first glance share with ESE the five Dublin Core properties Creator, Description, Rights, Subject and Title. However, the IPTC specification restricts the use of all these properties, often explicitly in disjunction with other IPTC properties (*i.e.* IPTC implements a specialised 'profile' of DC), which therefore cannot be mapped to the ESE schema without considering the specialised semantic value of the data. Where a photograph depicts another artwork, for example, there is no way to make this distinction in ESE to respect all the rightsholders and provide usable information.	Though a very small number of IPTC properties could potentially be mapped to ESE with agreement from IPTC as a "standard" mapping. The effort involved would be more efficiently spent creating individually tailored mappings for individual contributors as described for ONIX.
DDEX	The DDEX schema is of similar descriptive complexity to ONIX, but with the added structural complexity of contents lists for each "release". These may be impossible to map within LIDO, and thus certainly impossible with ESE.	The comments above on ONIX will apply here, given the similarity between the design of ONIX and DDEX, and the higher complexity of some DDEX structures. The initial mapping of DDEX to LIDO produced excellent ESE, but this points to the strength of the DDEX-LIDO and LIDO-ESE mappings, rather than the usefulness of a DDEX-ESE mapping.
EIDR	The EIDR schema is essentially a minimum referent data schema, and thus is relatively self-contained. It therefore offers the most promise for a more stable mapping to ESE, although the lack of domain-specific audio-visual content description elements in Dublin Core will probably mean this would be extremely minimal.	As for DDEX above, the EIDR-LIDO-ESE pipeline so far works very well; this could be used to derive a stable EIDR-ESE mapping, but since EIDR data can only come from one provider (EIDR itself) this seems inefficient.

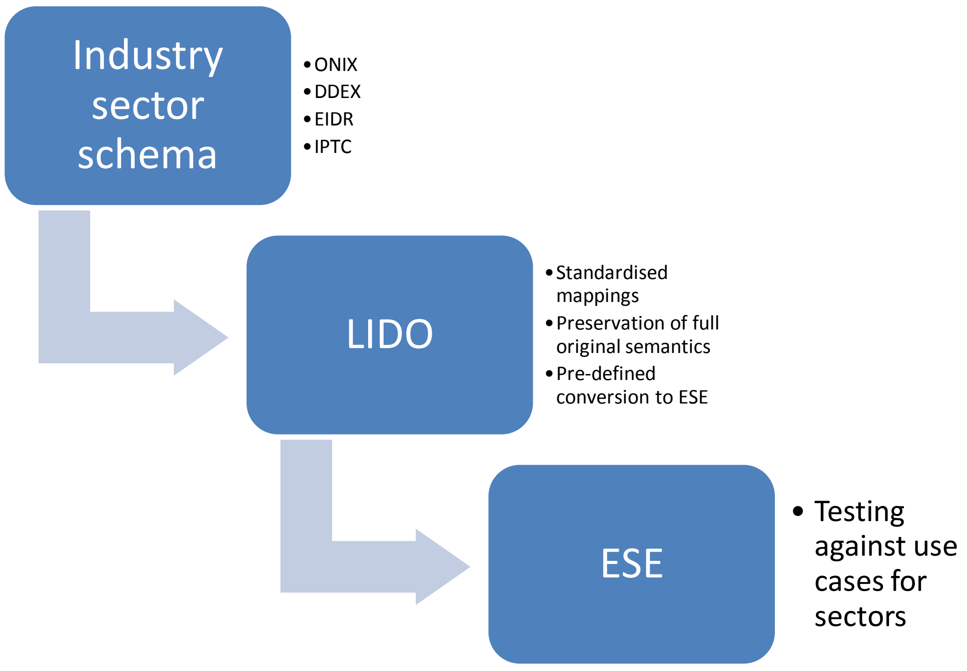
The experience of mapping ONIX and IPTC to ESE and the results of initial LIDO mappings showed conclusively that despite apparent mapping simplicity, this a false economy because it creates far more problems in the areas of legal-commercial agreements and the capabilities of the aggregation platform to apply business rules. Having attempted the first two mappings, therefore, attention was focussed exclusively on creating truly standardised semantic mappings to LIDO.

5.2 MID-COMPLEXITY MAPPING – ESE VIA LIDO

The core result of T4.4. is therefore the semantic mapping of ONIX 3.0.1. to LIDO 1.0., with similar mappings of ONIX 2.1., DDEX, EIDR and IPTC to follow. This mapping decision is in line with the "intermediary aggregator" approach taken for other Linked Heritage partners, as well a significant number of other projects contributing to Europeana[74]. This will provide the same benefits as aggregating cultural heritage data this way:

- Preservation of full, accurate semantics (for elements that can be included in LIDO);
- Preservation of (most) data granularity;
- Stability of mapping (in the face of changes within the Europeana data model);
- Separate control of LIDO database.

As noted above, the provision for retail links currently available in LIDO and MINT is not optimal; however, a minimum can be offered and the full cultural value of the product data aggregation should also be achieved.


Industry sector schema
• ONIX
• DDEX
• EIDR
• IPTC
LIDO
• Standardised mappings
• Preservation of full original semantics
• Pre-defined conversion to ESE
ESE
• Testing against use cases for sectors

Mid-level complexity mapping- industry schemas to ESE via LIDO

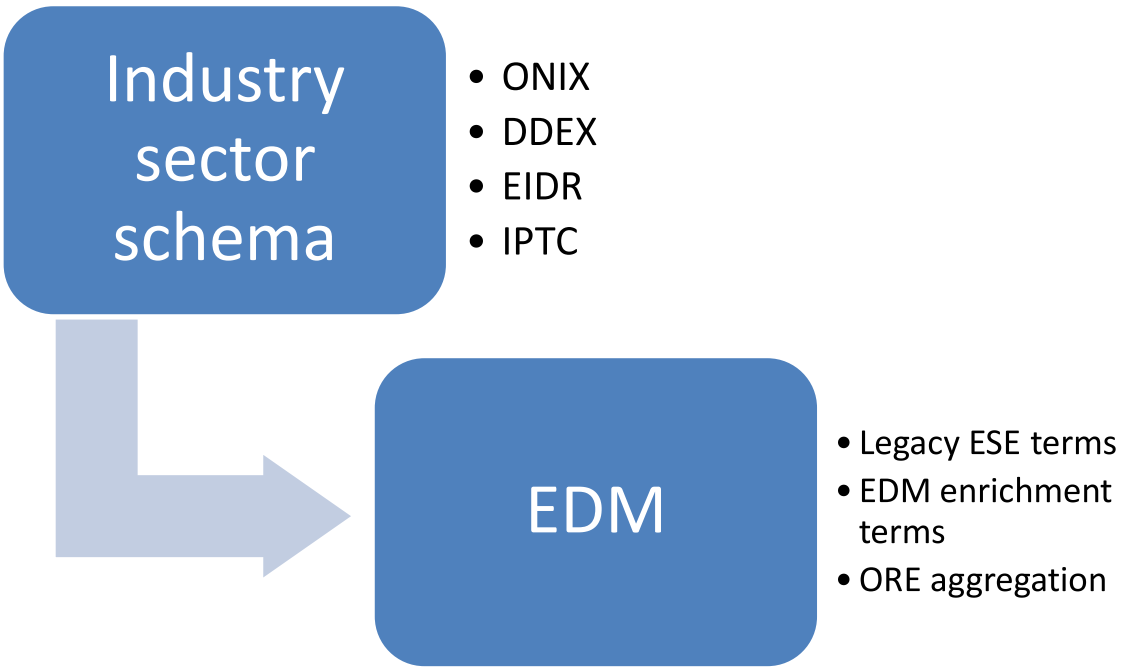
The above schematic for this mapping complexity level shows another decisive benefit – full respect for the existing best practice in semantic mappings by creation of agreed, standardised mappings between two standard schemas of well-defined semantics. A final benefit is that the existing MINT aggregation pipeline can be used to test the full data supply and commercial contributors can rely on the expertise and support of the Linked Heritage partners as well as that of Europeana. It is expected that this testing will take the form of "prototype" uploads (see section 15.2.2 and Appendix 5) of real commercial data feeds through MINT/LIDO into Europeana/ESE, so that further development of the legal-commercial framework can be undertaken by gaining feedback from the contributors, collecting statistics, and gathering any further technical requirements for the aggregation process itself.

There is not yet a standard mapping of LIDO to EDM in use within Linked Heritage so this option was not considered. However, it would not differ significantly from the approach described above, or the one described in the next section.

5.3 MAPPING TO EDM – BENEFITS AND CHALLENGES

Another possibility considered but, because of time constraints, not fully explored, was a direct mapping of industry standard schemas to EDM.

[74]See the list at http://mint.image.ece.ntua.gr/redmine/projects/mint/wiki/Projects



Alternative mapping option – industry schema direct to EDM

The main problem with this mapping approach would be that EDM inherits all the same problems as ESE, since this is the core of the EDM schema. The improvements inherent in EDM, which are largely concerned with events and relations, are already realised in the LIDO schema. The added structures for dealing with alternative surrogates for a single Cultural Heritage Object are interesting, but the use case for dealing with multiple views of a commercial product is very different (see section 5.1. and Appendix 2) and is not dealt with by EDM. It is worth noting that a direct mapping of EIDR to EDM would, as for ESE, bring the most benefit as the extra EDM relation elements could potentially express more of the EIDR semantics; however, as noted in section 3.1., it would still be minimal and would apply only to data exchange agreements directly between the EIDR registry and Europeana.

5.4 MAXIMUM COMPLEXITY SOLUTION – EXTENSION OR NEW SCHEMA?

The final option to consider would be to extend LIDO, as has been discussed in the context of FRBRoo, or use an even more general data model, perhaps generated through the VMF. This solution would be an ideal option except that its complexity means that it would require far more time and resources than currently available.

5.5 COMPLEXITY LEVEL CHOSEN FOR THIS REPORT

Having explored several of the available options, Work Group 4 decided to focus on standard, agreed mappings of industry sector schemas to LIDO, primarily the ONIX for Books 3.0.1. and 2.1. schemas, but with initial work on the other three schemas so that at least initial semantic mappings in each area could be available for testing on real data within the timescale of the project, and so that work on ESE mappings could be correctly placed within the business case development work of D.4.3. Knowing that best practice indicates compatibility with FRBRoo or VMF, and a stronger representation of rights data, recommendations were also developed for extending LIDO at a later stage through minor revisions to the existing version of the standard.

6 TECHNICAL SPECIFICATION – LIDO MAPPINGS

Although mappings of all four sector schemas were attempted, the ONIX for Books mapping was the only one ready for publication by the date of this report. This was because:

- It is the only schema mapping mentioned explicitly in the Description of Work and was thus accorded priority status;
- It is within the direct expertise of EDItEUR as the standards body that maintains and develops ONIX;
- Large amounts of sample data are easily available to create and test the mapping;
- It is complex and rich enough to represent the full range of semantic and technical problems relevant for the commercial schema-to-LIDO mapping landscape.

6.1 ONIX FOR BOOKS 3.0.1 MAPPING AS EXEMPLAR

The experimental mappings from the industry sector schemas to LIDO took the form, described above, of the normal Linked Heritage mapping work, mostly within the MINT tool and based on instance data, with the addition of a detailed comparison of the semantics and syntax of the schemas themselves, as well as the attempt to create an XSLT transformation for the full schema, even when instance data did not use every element of the industry sector schema.

Creating these mappings (although only the ONIX 3.0.1 mapping is as yet fully specified) had several benefits reported here:

- Achieving the core objective of representing ONIX for Books data in LIDO;
- Testing the ONIX to LIDO mapping with instance data and appraising suitability of the current LIDO to ESE mapping for this purpose;
- Allowing for a full exploration of the semantics and syntax of LIDO and its capacity to represent product types as well as unique individuals;
- Practical exploitation of the functionality of MINT and an appraisal of its strengths and areas for potential further development;
- Comparison of LIDO and MINT's capabilities for aggregating a wide range of media resources across all four media sectors.

The ONIX for Books 3.0.1 mapping to LIDO is described here in detail as an example of a complete LIDO mapping from the commercial sector. Findings from the ONIX mapping were found to apply generally across all four sectors since the ONIX standard is highly developed and incorporates all the essential features of commercial product data; some other findings were sector specific and are reported separately.

6.2 PRESENTATION OF MAPPINGS

Outline discussion of the LIDO mappings is presented here in the main body of the report, so that although technical accuracy is conveyed, less specialised knowledge of XML, XSLT and the details of the LIDO and ONIX for Books standards will be required. For readers interested in the detailed structure of the ONIX mapping, it is presented here as a full XSLT listing plus commentary in Appendix 3, and also as an equivalent but easier to read mapping syntax in the separate Excel spreadsheet uploaded to the Linked Heritage website alongside this report.

6.3 DOCUMENTING SEMANTIC MAPPINGS

One of the apparent ironies of the current applied research scene with respect to data integration and Web-enabled data is the prominence of references to semantics despite the seeming lack of detailed discussions of semantic mappings understandable by the moderately technical, non-domain expert reader. Of course, many such mappings are thoroughly documented, but the verbosity and complexity of their expressions in languages such as XSLT make them unwieldy for readers and almost impossible to present in full while preserving their significance.

Here, a combination of approaches has been taken, to enable a reader-friendly narrative that gives sufficient technical detail to make the report usable for Linked Heritage and as a starting point for further research and development. The main approaches are:

- Presentation of large, complex schemas in outline at a low level of detail, giving a heuristic overview of the main schema "entities"; usually this means presenting "target right" to show the framework *into which* the source schema will be analysed for re-expression;
- Description of detailed semantic mappings – statements of equivalence – presented "target left" so that a readable, narrative-style formulation is available (though still following the logical order of the *target* schema as this is the "language" which provides the "context" of the mappings);

6.4 MAPPING SYNTAX USED IN ACCOMPANYING SPREADSHEETS

The ONIX 3.0.1 mapping made available along with this report (as for all spreadsheets of mappings still to be released) uses a simplified syntax to describe semantic equivalences and the XSLT syntax used to express these. It follows the actual mapping decisions made in the MINT aggregator and thus is a translation of XSLT, but is suitable for non-specialist readers. The syntax is as follows:

Notation in spreadsheet	Explanation
Map this pair if this XPATH…	Condition source element
Exists	Element present in source XML?
>	Value is more than…
<	Value is less than…
=	Same as…
…in this namespace…	ONIX code list [x]
AND	Logical operator to link to row directly below
OR	
NOT	
…in preference order…	Order of preference of several mapping options listed directly below
…and use this constant value (or Code List map) in the target XPATH	Value to be used either a) to produce a constant output value, or b) compare with the source element value according to specified value operator (above)
+	Concatenate value directly below
&	Map value below to a new target element

7 LIDO AS A TARGET SCHEMA FOR PRODUCT DATA

Here a general outline of the LIDO schema itself is described, forming a narrative structure for the mappings to follow and giving readers new to LIDO an informal but accurate idea of its logical structure.

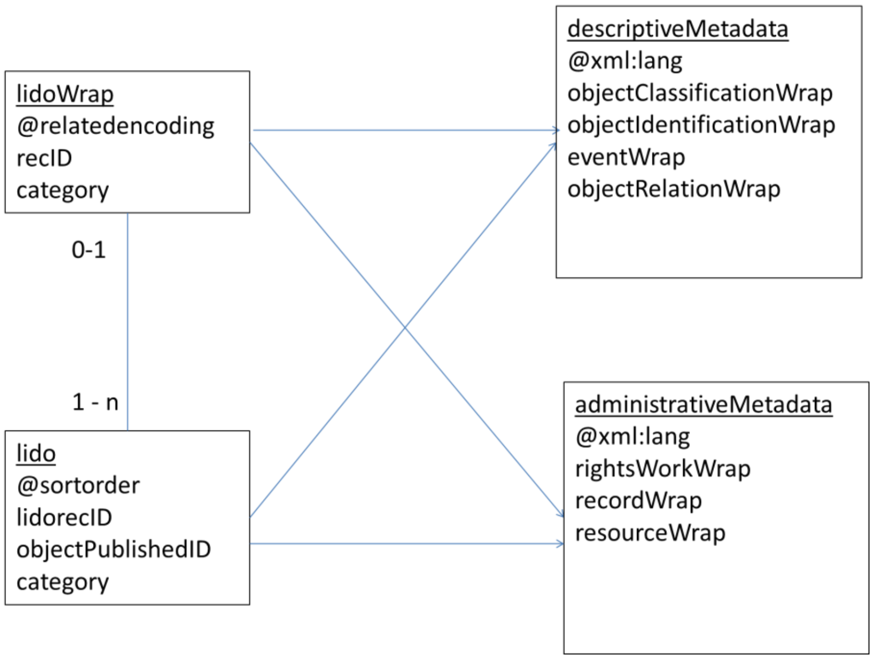
At each stage general points that count for and against LIDO's suitability as an integration format for commercial product data are noted. These apply across all four media sectors.

7.1 LIDO SCHEMA OUTLINE

The LIDO schema has a flexible top-level structure that optionally allows one LIDO file to carry any number of object records:

LIDO	Comment
`<lido:lidoWrap>`	Optional "wrap" to contain multiple lido records
`<lido:lido> … </lido:lido>`	LIDO record #1 [subheadings hidden for clarity]
`<lido:lido> … </lido:lido>`	LIDO record #2
`<lido:lido> … </lido:lido>`	LIDO record #3… etc.
`</lido:lidoWrap>`	

The optional multiplicity of records in one LIDO XML document is summarised in a different way in the structure diagram below. Note that the diagram simply shows the cardinalities of the subelements in the schema hierarchy and the attributes attached to each element; it is not a full UML class diagram.



Top level structure of LIDO documents (simplified hierarchy and cardinalities)

When <lidoWrap> is not present, a single <lido> element instead forms the root node of the XML document *i.e.* the file only contains one object record. Two of the source schemas, ONIX and DDEX, have a similar top-level structure (one message containing multiple "product" or "release" records) and so the root and item level nodes can be matched easily within LIDO and MINT. For the other two schemas, EIDR and IPTC, the situation is more complex, but source files can be pre-processed with relative ease to achieve the same result, matching item nodes to either <lidoWrap> or <lido> at the convenience of the data provider.

Within the item record defined by the <lido> element, the structural and data elements are then broken down into descriptive and administrative types, further subdivided as shown below. In the following tables, XML elements are shown nested as in an actual instance data file; empty nodes are shown both opening and closing where the <element /> in question is fully enclosed by its superelement.

LIDO	Comments
`<lido:lido>`	
`<lido:lidoRecID/>`	Identifier for this LIDO record.
`<lido:category/>`	The type of CIDOC-CRM entity described by this LIDO record. For product data, always F3 Manifestation Product Type.
`<lido:descriptiveMetadata>`	
`<lido:objectClassificationWrap/>`	Use of controlled vocabularies to classify objects; effectively the same as for products.
`<lido:objectIdentificationWrap/>`	Information that distinguishes this object from others in the same class. Most is similar for products except for two areas unique to individual object (see detailed breakdown in following sections).
`<lido:eventWrap/>`	Events will be taken from an object's "life history" or a product's "life cycle". The event structure allows the decomposition of data from many different "flattened" structures and integration into one database (as in VMF)
`<lido:objectRelationWrap/>`	Relations allow links between object or product records to be established and assigned types (classifications).
`</lido:descriptiveMetadata>`	

LIDO	Comments
`<lido:administrativeMetadata>`	
`<lido:rightsWorkWrap/>`	Rights relating to the object itself. For products, broadly the same.
`<lido:recordWrap/>`	Information about this LIDO record's source data. Includes link to DO in context.
`<lido:resourceWrap/>`	Information about digital representations of the CHO, including the DO. For most products, analogous to CHO resources but for photos, could refer to different versions or even related products.
`</lido:administrativeMetadata>`	
`</lido:lido>`	

The <lido/> encloses two sections, dividing the content into "descriptive" and "administrative" metadata[75], as well as a small number of initial elements applying to the "whole record". These are described in the next sections.

7.1.1 LIDO "whole record" elements

LIDO	Comments
`<lido:lido>`	The containing element for the whole object record
`<lido:lidoRecID/>`	An identifier for the LIDO record itself. At least one record ID is normally present in product data.
`<lido:category/>`	The type or scope of the LIDO record; recommended to be taken from the CIDOC-CRM. For commercial products this should always be set to F3 Manifestation Product Type. See the discussion in section 5.1 and Appendix 2 for the justification.

[75] This follows the typical classification found in most discussions of metadata (*e.g.* http://www.niso.org/publications/press/UnderstandingMetadata.pdf as cited in D4.1). LIDO does not appear to have been based on the principle that "all data are rights data" (see http://www.dlib.org/dlib/july98/rust/07rust.html) and more significantly, this structure is a closer fit for the "static repository" model of data exchange (see D4.1, section 5.3).

7.1.2 LIDO Descriptive Metadata – Classification

LIDO	Comments
`<lido:descriptiveMetadata>`	Wrapper for the descriptive elements.
`<lido:objectClassificationWrap>`	Wrapper for the two levels of classification specified in LIDO.
`<lido:objectWorkTypeWrap/>`	The "Work Type" is defined as the most specific class that applies to the entire object; thus it is a subset of the classifications below.
`<lido:classificationWrap/>`	All other classifications that can be applied to the object; specifically, those that are described with controlled value lists. Both this and the objectWorkType sets take pairs of label ("term") and concept identifier.
`</lido:objectClassificationWrap>`	

7.1.3 LIDO Descriptive Metadata – Identification

Those in **bold** are the areas where the LIDO schema is particularly unsuitable for use with product types as they are not necessarily directly inherited by the type from its instances (or product exemplar).

LIDO	Comments
`<lido:objectIdentificationWrap>`	
`<lido:titleWrap/>`	Titles for the object (product)
`<lido:inscriptionsWrap/>`	Text appearing on the object
`<lido:repositoryWrap/>`	**The physical place and organisation of custody of the object – for products there is none**
`<lido:displayStateEditionWrap/>`	**Details of the "state" of completion of the "work" represented by this unique item – *e.g.* a stage in production or an edition**[76]
`<lido:objectDescriptionWrap/>`	Descriptive notes; found in all commercial schemas

[76] See a full explanation of "state" in heritage terminologies at: http://www.getty.edu/research/publications/electronic_publications/cdwa/5state.html - although this is one aspect of heritage object description not possible for (most) product classes, it hints at an identification of work types already present in LIDO.

LIDO	Comments
`<lido:objectMeasurementsWrap/>`	A generalised set of measurements (dimension, value and units) with added qualifiers to specify the aspect of the object being measured
`</lido:objectIdentificationWrap>`	

7.1.4 LIDO Descriptive Metadata – Events

LIDO	Comments
`<lido:eventWrap>`	
`<lido:eventID/>`	LIDO views “events” in the context of CIDOC-CRM’s interest in documented historical events; hence these entities can be identified for linking and comparison. The classes of events allowed currently in LIDO (see eventType below) naturally reflect those in which objects were the subject of the event but in principle need not[77].
`<lido:eventType/>`	The LIDO specification comes with an event type list (see section 4.4.5) based on that found in CIDOC-CRM and therefore compatible with the FRBRoo analysis.
`<lido:roleEvent/>`	Since this field does not yet have an assigned controlled vocabulary, in principle it could take a wide variety of values; in practice, for objects and works, it is likely to assume the value of the passive voice of the eventType.
`<lido:eventName/>`	Historical events are very likely to have names and titles; in commercial metadata, this is less important, if at all.
`<lido:eventActor>`	The actor information is likely to be very similar in both heritage and commercial contexts since the basic scenario is the creation and publishing of a creative “work”, whether in one or a class of many physical items.
`<lido:actorID/>`	Note that here, the identifier is for the actor themselves, whereas the main identifier for public *personae* in the commercial sector, the ISNI, is for names.[78]
`<lido:nameActorSet/>`	As noted above, in commercial schemas, the ID above would be linked to one or more variants of a name, rather than an independent data field.
`<lido:nationalityActor/>`	Places of birth and death are likely to be relevant for both heritage and commercial identification.

[77] See for example the commonly used CIDOC-CRM Core example describing the Yalta Conference: http://www.cidoc-crm.org/crm_core/core_examples/yalta.htm
[78] See D4.1 section 6.2.1 and also section 9.5.7 of this report for a discussion of name *versus* person identifiers.

LIDO	Comments
`<lido:vitalDatesActor/>`	As above, birth and death dates are relevant to both sectors, although for some public identities, they may count as private information in the commercial sector.
`<lido:genderActor/>`	Gender is unlikely to be found in commercial sector data.
`</lido:eventActor>`	
`<lido:culture/>`	The description of a work by its originating culture is highly specific to the heritage sector[79] but could potentially be found in some commercial data where the content has a general "cultural" or "national" aspect (*e.g.* published recordings of ethnic music, textual compilations of oral traditions, or photographs of national dress).
`<lido:eventDate/>`	The date is fundamental to identifying events in both sectors.
`<lido:periodName/>`	Again, the use of named time spans is specific to cultural heritage[80], but as for "culture", may be present in heritage publications where it will represent the subject matter (or possibly, by analogy, the style of a replica – see Appendix 3).
`<lido:eventPlace/>`	As with the date, a fundamental identifier for any event in both sectors.
`<lido:eventMethod/>`	Further qualifies the activity in eventType; found in both sectors.
`<lido:eventMaterialsTech/>`	Mainly of interest in the heritage sector, but again, potentially used for commercial products where the material (*e.g.* of the pages or binding of a printed book) is of interest.
`<lido:thingPresent/>`	A generalised reference to another object involved in this event; potentially interesting for both sectors but probably uncommon in daily use in commercial data.
`<lido:relatedEvent/>`	A generalised related event entity is unusual in commercial schemas, since they do not attempt to portray historical narrative. For the purpose of decomposing a complex term or expression by mapping into LIDO, this structure could potentially be used but this would require significant extra work from both sectors.
`<lido:eventDescription/>`	Descriptive notes may be found in both sectors, but are more likely to be qualified by limitation to one aspect of an event in commercial schemas.
`</lido:eventWrap>`	

[79] See the CDWA notes for examples of heritage usage: http://www.getty.edu/research/publications/electronic_publications/cdwa/14creation.html#culture
[80] See CDWA discussion of period and style: http://www.getty.edu/research/publications/electronic_publications/cdwa/17styles.html

7.1.5 LIDO Descriptive Metadata – Relation

LIDO	Comments
`<lido:objectRelationWrap>`	
`<lido:subjectWrap/>`	A subject in LIDO can be a simple "concept" (*i.e.* an entry from a classification scheme) or an entity (place, actor, date, event or object). Detailed subject information is only found in two commercial schemas (ONIX for Books and IPTC), even though for the other two (DDex and EIDR) it could be provided (perhaps through links to another source). In any case, the LIDO structures cover the full range of subjects found in commercial data.
`<lido:relatedWorksWrap/>`	The section in LIDO for related works is a complete generalisation allowing any other class of relation than "subject". This is present in some commercial schemas, and normally a type of relation is specified.
`</lido:objectRelationWrap>`	
`</lido:descriptiveMetadata>`	

7.1.6 LIDO Administrative Metadata – Rights Work

LIDO	Comments
`<lido:administrativeMetadata>`	Wrapper for administrative metadata.
`<lido:rightsWorkWrap/>`	A "right" set in LIDO is a basic structure composed of a type, date and rightsholder. This is significantly simpler than most rights in commercial data, which very often depend on territories, markets, relative publication and release dates of other products, and uses made of the products described; not to mention the nesting of rights within a single product due to the nature of collaborative, multimedia, or performance- or recording-based works. It should be noted that with an expanding scope, LIDO may begin to describe precisely such works in current or future projects (see sections 4.3 and 15.2.5).

7.1.7 LIDO Administrative Metadata – Record

LIDO	Comments
`<lido:recordWrap/>`	Describes the source record in terms of ID, source, type and associated rights. Most of these are present in some form in commercial data, except for a rights statement. LIDO record data also includes the recordInfoSet which identifies a public version of the same source record used to produce the LIDO; the URL here (recordInfoLink) is used for the Linked Heritage and Europeana use case of providing the digital object in context (europeana:shownAt). The product in context link is rarely found directly in product information data even if the schema allows it (as ONIX for Books does) because it is commercially sensitive information.

7.1.8 LIDO Administrative Metadata – Resource

LIDO	Comments
`<lido:resourceWrap/>`	This section is used to hold information about the Digital Object (see section 3.1) in the Linked Heritage and Europeana aggregation context. The LIDO specification states that this section excludes "items that are considered objects / works in their own right", a problematic view since, by European law, and in the commercial perspective, this would exclude all resources, as even an informal personal photograph can be considered a creative work for copyright purposes. The LIDO specification seems to implicitly acknowledge this by providing a rights section (details below).
`<lido:resourceID>`	The identifier for the original resource.
`<lido:resourceRepresentation>`	Contains a URL and measurements of different sized versions of the same image file.
`<lido:resourceType>`	A broad classification of the genre of the image, rather than subject matter or technical format.
`<lido:resourceRelType>`	Rather than describing the "relationship" of the image to its subject in terms of recording process (as resourceType does) this actually records the purpose or context for taking the image.
`<lido:resourcePerspective>`	This applies above all to physical items and is unlikely to appear in creative media product data (even if it could technically appear in a photo product description, it is not found in the IPTC vocabularies explicitly).
`<lido:resourceDescription>`	Simple descriptive note often found in commercial data.
`<lido:resourceSource>`	This and the field below are essential data for commercial use.
`<lido:rightsResource>`	As mentioned for the rights fields above (section 8.1.6) this is a far simpler expression than is normally found in most commercial data. See in particular the discussion of photo rights in sections 12 and 15.2.5.

LIDO	Comments
`</lido:administrativeMetadata>`	
`</lido:lido>`	

7.2 LIDO ATTRIBUTES

The use of XML attributes in LIDO mainly follows the design principle mentioned in section 4.4.3; they are used to constrain the encoding or semantic category of the data in the elements they are attached to. However, due to the LIDO's use as an aggregation format, and its bias towards heritage objects, very few of these elements have analogues in commercial data. Sometimes this is unproblematic but a small number of examples may lead to difficulties in aggregating both commercial and more complex heritage data, primarily because use of attributes prevents delivery of multiple values.

LIDO attribute	Similar attributes or elements in commercial schemas?	Comments
@addedSearchTerm	None	Used in aggregation to distinguish terms meant for record retrieval only. This has a small number of equivalents in commercial sector data, for example, product titles used only by one part of a supply chain.
@codecResource	Normally provided as elements to give details	Codec information is given in more detail when a digital resource forms the main content of a product.
@encodinganalog	None	Used in aggregation to represent the source schema's field for the same data. Not present in commercial data (although IPTC's use of external namespaces is similar) but equivalents can be found in mapping tools such as VMF.
@formatResource	Normally provided as elements to give details	Internet MIME types for resource format are inappropriate for most commercial schemas, although sometimes used.
@geographicalEntity	None	Geographical location is not normally specified in product metadata (although it could occur *e.g.* in a subject scheme, especially for cartographic products).
@label	None	Used in aggregation to capture field labels for display; not used in product data where a label is more likely created by the data's end user.
@politicalEntity	Some	Normally part of an element definition of *e.g.* country of manufacture, city of publication, sales right territory.
@pref	Some	Normally present as a "flag" element or binary data value.

LIDO attribute	Similar attributes or elements in commercial schemas?	Comments
`@relatedencoding`	None	Used in aggregation to denote the namespace from which source element (field) names are taken. Can be extracted from commercial schemas expressed as XML schema definitions. The relationship between the namespace captured here and the identifiers for the schema's elements (mapped to the @encodinganalog attribute) follows the same pattern as the elements of a controlled value set or SKOS concept scheme, and could potentially be managed using the Linked Heritage TMP.
`@sortorder`	Some	Often used in specific circumstances in commercial schemas but expressed in very different ways (*e.g.* as an RDF sequence in IPTC; as XML values in ONIX for Books).
`@source`	Normally provided as elements to give details	References a controlled value set for the element in LIDO; in commercial schemas normally more detail is needed; for example, the version of the vocabulary, or the name of a proprietary classification.
`@type`	Normally provided as elements to give details	Because type vocabularies depend heavily on the data in question, and its use case, in commercial data this is expressed in more complex ways than the single attribute available in LIDO.
`@xml:lang`	Some	The XML language attribute is problematic because of the complex controlled value set used to populate it. In commercial schemas (*e.g.* DDex) it can appear, but often a schema-specific language element is used for simplicity.

7.3 SYNTACTIC-SEMANTIC ASPECTS

Here we will note some features of LIDO found across the entire schema, where the schema's syntactic structure, somewhat reflecting its conceptual basis in CIDOC-CRM, affects the semantics that can be expressed.

7.3.1 Object and resource separation

The LIDO schema allows far more details to be recorded about the CHO than about its digital representation (the DO). This appears to be a feature inherited from CDWA Lite[81]. In other cultural heritage schemas, notably the VRA Core schema[82] it is possible to specify as much information about the resource representing a cultural work, as for the cultural work itself, effectively treating the DO as a CHO in its own right. Of course, VRA Core is still far less detailed as a whole than LIDO; the key difference is that more culturally relevant details can be added to the DO part of the record (actually in VRA it would be a full record each for the DO and CHO, linked through identifiers).

For books, music and film data this is probably unproblematic. The image of a book, CD or DVD cover, while certainly a creative work in its own right, for these purposes is used primarily in a compressed form and treated as marketing collateral. In the case of commercial photographs, the DO is certainly to be considered as important as any CHO it depicts, as the digital photography file is itself the "product". This may mean that LIDO's treatment of digital resources is not sufficiently detailed.

7.3.2 Event structure

As explained in section 4.4.6, the event or "contextual" approach to metadata is the most expressive and allows practically any type of data to be integrated. The LIDO schema incorporates an event structure explicitly, which, although specialised somewhat for historical museum object description, can be considered general enough for integration of basic event data from any domain.

Events in a product's lifecycle often appear through "flattened" semantics in product data (although notably DDex contains event-like composites for dates, and many parts of ONIX are full or nearly full event structures). Extracting the relevant parts of the event information into LIDO will allow for integration of data from other sources to create more culturally valuable data sets and links.

7.3.3 Internal and display elements

The nature of LIDO as aggregator schema is apparent in its separation of "internal" data for search and retrieval from "display" data (normally at a "set" level in the schema) and the "label" attributes that can be attached to most elements (see above, section 8.2). This type of separation is only partially realised in commercial product data, where much of the data may be "raw" information for use within the supply chain, or else require significant processing to recompose it in an intuitively comprehensible form for end-users. Hence the display elements in LIDO may be more useful for capturing "alternative" free-text data elements from otherwise complex source schema sections, and the labels may be better used for relating data values to their original schema and best practice to aid implementation of a user-facing display (see notes on ESE display in section 6.1 and recommendations in section 15.2.1).

7.3.4 Appellation Values and Sources

The LIDO schema's structure is partly derived from the CIDOC-CRM, which reflects museum documentation practice. A core value of this practice is the documentation of "appellations" (names, titles and other labels)

[81] See the history of LIDO's development here: http://network.icom.museum/cidoc/working-groups/data-harvesting-and-interchange/lido-overview/lidos-background/.
[82] See the VRA Core 4.0 introduction here: http://www.loc.gov/standards/vracore/VRA_Core4_Intro.pdf

and their “sources” in harmony with historical methods of citation, and to allow multiple viewpoints on each object. This approach is foreign to the use of commercial schemas, where each distinct part of a product description is ideally an integral product of one reliable supply chain partner[83]. Thus the source element will not be often employed for aggregating this data. On the other hand, various types and connected parts of titles and names are almost always supplied in commercial data for use in different contexts; this does not appear to be supported by LIDO’s name model.

7.3.5 Concept IDs and Terms

Finally, LIDO extensively uses a another element pair consisting of a “concept ID” and free-text “term” to provide concepts from controlled value sets when these are part of the core source data, as often happens in commercial data. The correspondence is only partial, as when these pairs must be mapped to a LIDO “type”, only a term can be used since LIDO (mostly) expresses types with the @lido:type attribute.

[83] See discussion of metadata use cases in D4.1 section 5.3.4.

8 ONIX FOR BOOKS 3.0.1 AND 2.1 MAPPINGS

This section details the entire semantic and syntactic mapping of the ONIX for Books version 3.0.1 product information message to LIDO. In the final section details of the related mapping of the previous (and most widely used) version, ONIX for Books 2.1, are given. Because ONIX 3.0.1 is the most up-to-date version, and includes such a comprehensive range of the features exhibited by commercial data schemas generally, it proved an excellent proxy for commercial data in general for the purposes of this exercise.

8.1 CONDITIONS FOR INCLUSION OF ONIX RECORDS

Key XPATHs (see section 4.4.3 and glossary in Appendix 1 for this term) within an ONIX message specify an aspect of the product that is crucial for that product's inclusion or not in Linked Heritage or Europeana. These are listed below, although note that the first criterion relates to wider issues of the legal-commercial framework (selection of data that can be acceptably supplied to all end customers) and the data model and software platform (filtering of data according to *e.g.* territory, absolute or relative release dates).

ONIX XPATH All begin with ONIXMessage/Product/	Allowed values for inclusion of record	Meaning
…RecordSourceType	01 (other values may be acceptable)	Indicates which partner in the product supply chain is the source of this record. Could be a convenient way to select only records that originate directly from the publisher as the "repository" of this product.
…NotificationType	03	Indicates a complete record for a book already or "soon to be" published. Thus it should be available to retail customers.
…PublishingDetail/PublishingStatus	04	The product is "active" and can be ordered from the publisher
…DescriptiveDetail/ProductComposition	00 or 10	Indicates a product meant for retail.
…DescriptiveDetail/TitleDetail/TitleType **[and other conditions – see the restrictions on titles that can currently be mapped to LIDO in section 9.5.3]**	01	The product record provides a "distinctive title" for the product, to map to the mandatory LIDO elements in lido:titleSet.

These conditions specify the classes of ONIX records that should be included; implementation of these rules currently would have to be done by data contributors themselves, or as part of a pre-processing stage before aggregation in MINT.

Note that since this part of the specification touches on agreements made with data providers, it remains to be addressed in D4.3.

8.2 ONIX CODE LISTS

The ONIX code lists were included in the XSLT mapping as variable "maps" like the one below for product identifier types (code list 5)[84]:

[84] Full ONIX code lists: http://www.editeur.org/ONIX/book/codelists/current.html

```
<xsl:variable name="map0">
    <map value="GTIN-13">03</map>
    <map value="UPC">04</map>
    <map value="ISMN">05</map>
    <map value="DOI">06</map>
    <map value="LCCN">13</map>
    <map value="GTIN-14">14</map>
    <map value="ISBN">15</map>
    <map value="Legal deposit number">17</map>
    <map value="URN">22</map>
    <map value="OCLC number">23</map>
    <map value="ISBN">24</map>
    <map value="ISMN">25</map>
    <map value="ISBN">02</map>
  </xsl:variable>
```

This "map" replicates the code and label (SKOS:notation and SKOS:prefLabel) columns of ONIX code list 5 and allows MINT to complete the @type attribute for <onix:IDValue> elements wherever they are present in the input ONIX file. Since they can occur in many places, such as the published identifier for the product of interest itself, related products and parts of products (which are products in their own right), the code list also appears again in maps 138 and 156 (their numbering is in multiples of 2). A simple optimisation of the XSLT code would be to re-use the same XSLT variable in every instance instead; this is not yet possible in MINT.

This is a necessary duplication at present since there is no other way to refer to the code lists. It introduces both redundant code, and the need to change the XSLT each time code lists are updated. One simple improvement to the existing MINT software would be to allow custom names for the "maps" to link them to their source data, so that mapping and schema owners such as EDItEUR could track them automatically even if updating them manually.

It would be more efficient to refer to them using SKOS, and indeed this is planned by the Linked Heritage terminology group. Replacing value maps like these with indirect references to a SKOS ConceptScheme would also solve the problem of updates to the code lists; at present, since the codes are embedded in the mapping, the XSLT must be updated when the code lists change (quarterly); by using a URI reference to the current list, MINT could simply transform the codes into labels using the latest version each time a new ONIX record is uploaded.

There are 87 such "maps" in the full XSLT, and their structure is entirely predictable from that above and the relevant code list values and descriptions, so they have not been presented in Appendix 3 as part of the commented listing. A simple list of the "map" variables, with the code lists they correspond to, is included there instead.

8.3 ATTRIBUTE MAPPINGS (WHOLE LIDO RECORD)

A small number of LIDO attributes are used in a consistent way to map ONIX fields across the entire output record. These are as follows.

8.3.1 @type

The LIDO @type attribute has been used in a variety of ways to map ONIX elements. The most general ONIX elements using @lido:type were those representing dates, described entity identifiers, concept identifiers and titles.

- Dates

ONIX dates mostly carry a @dateformat attribute, and in most cases this has been mapped to @lido:type for a LIDO date element, using code list 55 to map the date format values (*e.g.* YYYYMMDD or YYYY). Exceptions are when the ONIX element is explicitly limited to one temporal term – mostly year[85]. Then the @lido:type is simply set to the equivalent text for that format, in the case of year, "YYYY".

The LIDO schema specification for date elements states "General format: YYYY[-MM[-DD]] Format is according to ISO 8601", though it is not clear if this means separators should always be used or not, and whether other ISO 8601 formats than YYYY[-MM[-DD]] are also acceptable. In any case, neither the LIDO XML schema nor MINT validate for this, and most ONIX date format options fall within the range of ISO 8601. The preservation of ONIX @dateformat should allow applications to properly use the LIDO dates aggregated from ONIX messages.

- Described entity identifiers

The ONIX elements IDTypeName and those whose reference names are suffixed -IDType are used in different contexts throughout the ONIX for Books message. The -IDType element takes a value from code list determined by the type of entity identified and takes a value from the relevant code list for that context. In the table below, these are listed in XML document order – note that most use is made of name and product identifiers, and that the latter part of the ONIX message, where these are less common, is not mapped to LIDO:

Entity class	ONIX –IDType elements	ONIX message context(s)
Persona[86] or organisation name	<SenderIDType> <AddresseeIDType> <RecordSourceIDType> <NameIDType > <ConferenceSponsorIDType> <ImprintIDType> <PublisherIDType> <ProductContactIDType> <CopyrightOwnerIDType>	Header Product record Contributor Conference [not mapped to LIDO] Publishing
Product	<ProductIDType>	Product record Product part Sales rights [not mapped to LIDO] Related product
Work		Related work
Collection[87]		Collection
Text item		Content [not mapped to LIDO]

[85] ONIX elements typically carrying only YYYY-format dates: YearOfAnnual (also spread of years, but this is not supported in LIDO), ThesisYear, PrizeYear, CopyrightYear. YearOfAnnual is the only ONIX "year" element that actually allows free-text input, although the specification specifically states that it should typically hold "years" (not, therefore *e.g.* seasons, months, *etc.*).

[86] See section 9.5.7 for discussion of data models for names.

[87] In the heritage context, a "collection" is a set of individuals, two or more physical objects that could be (and probably are) found in one location. Although the connection between the objects is an abstraction (like any set) the collected objects are all unique items. In ONIX for Books, in contrast, a collection is an abstract set of product types; a double abstraction. See also the ONIX best practice note on sets and series: http://www.editeur.org/files/ONIX%203/ONIX_Books_Sets_and_Series_3.pdf

Entity class	ONIX –IDType elements	ONIX message context(s)
Supplier	<SalesOutletIDType>	Sales rights [not mapped to LIDO]
		Product supply [not mapped to LIDO]
	<AgentIDType>	Market publishing [not mapped to LIDO]

In each case where an ONIX entity identifier is mapped to a LIDO identifier element, the ONIX –IDType element is mapped to @lido:type attribute, using the relevant ONIX code list to convert the code values into a meaningful, human-readable ID type name. The only exception to this rule is where the –IDType element contains a code list value that specifies a proprietary identifier type; then the name of this identifier system is then found as plain text in the onix:IDTypeName element which will be mapped to the @lido:label attribute (see section 8.3.3 below).

- Concept identifiers

In most cases in the LIDO mapping, lido:conceptID elements take a @lido:type containing the value "local" because they are direct imports of an ONIX code list value which is "local" to ONIX messages. In rare cases such as subject classification schemes, the ONIX data values are references to external, published subject schemes, and there, the name of the scheme is used for the LIDO identifier type (see section 9.5.13 for discussion of this case).

- Titles

The @lido:type attribute was also found valuable for use mapping ONIX titles where there is a generic subdivision of title and subtitle.

8.3.2 @xml:lang

The LIDO language attribute has been used in two ways to map ONIX data:

- Where a data value will be taken from an ONIX code list, @xml:lang was set to "en" since the primary language of the code lists' concept labels is English. However, this mapping will not be necessary if and when SKOS code lists can be integrated into LIDO, since the language of the aggregated data could then be taken from the SKOS concept.
- Where a date value has a corresponding @onix:language attribute it maps directly to the @xml:lang LIDO element.

It must be noted here that this over-simplified mapping is contrary to the definition of @xml:lang[88] since the @onix:language attribute takes only ISO 639-2/B (three-letter) codes, whereas @xml:lang can have a mixture of two- and three-letter codes as its content[89] - hence any application using the LIDO data generated from this ONIX mapping must re-map the @xml:lang content to acceptable IANA-registered values[90]. Note also the recommendation on language code mappings in section 15.2.5 of this report.

8.3.3 @label

The @lido:label attribute has been used in two primary ways in this mapping from ONIX.

- To carry the source ONIX element's name precisely as given in the ONIX for Books 3.0.1 specification documentation (*N.B.* not the XML element name, but the natural language name used to describe the unique element in the context of its position in the whole schema). Note that similarly to the

[88] See http://www.w3.org/TR/REC-xml/#sec-lang-tag and http://tools.ietf.org/html/rfc4646 for the official definitions of language tags for the xml:lang attribute.
[89] See the W3C pages on language tags for XML for the full discussion of why the types of language codes are mixed: http://www.w3.org/International/articles/language-tags/Overview.en.php
[90] See http://www.iana.org/protocols/ for the full list of IANA-registered language codes.

@lido:encodinganalog case in the section below, this is a purely mechanical mapping, and thus has only been implemented here where it adds to the semantic value and use case of the LIDO aggregation (otherwise it simply increases the size of the XSLT script) and will only be fully implemented for specific test data sets.

- To carry a very specialised label for elements where this is specified in a code list – the most obvious example being the name of a proprietary identifier scheme as noted in section 9.5.13 above.

The natural language element name is appropriate for this LIDO element as it is specifically intended to carry some of the semantics of the data value to the end user of the data, to aid in interpretation of the data value. Any automated application of the element names should rely on the mapped XPATH from the XSLT, or unique ID number of the element (see section 9.3.4) below.

8.3.4 @encodinganalog

The @lido:encodinganalog attribute allows preservation of the source element name (or reference) within the aggregated LIDO record. In the case of schemas such as ONIX where an XML schema definition exists, each uniquely defined possible element can be identified by an XPATH, or some other unique identifier. Currently there is only one possibility for implementing this, using the ONIX schema element reference numbers beginning "H" for header elements and "P" for product record elements (see Appendix 5 for these). EDItEUR is considering releasing canonical HTTP URIs for the ONIX elements and these could potentially be used in future. Since this mapping is entirely mechanical and would increase the length of the XSLT mapping script it has not been included in the XSLT listing in Appendix 3 or the XSLT file accompanying this report, but will be implemented with the first test or prototype data set.

8.4 ELEMENT MAPPINGS – LIDO RECORD

For each part of the mapping presented here, examples of output LIDO elements will be shown, and the rationale behind the mapping method discussed. For the full XSLT, refer to Appendix 4, where the XSLT stylesheet is found in full, divided into the same sections as here.

8.4.1 Template – lidoWrap

The ONIX for Books XML message format maps perfectly to LIDO's flexible record/document structure, as implemented in MINT. When an ONIX file is uploaded, the MINT "root node" is set to the <ONIXMessage> element, and the "item root" is set to the XPATH for each ONIX Product Record: ONIXMessage/Product. This ensures that the arrangement of data records within a document in ONIX and LIDO correspond at the top level, one input product record mapping to one output object record.

8.4.2 Template - @relatedencoding

The LIDO @relatedencoding attribute is applied to the LIDO record element with a constant value specific to the ONIX 3.0 mapping:

```
<lido:lido lido:relatedencoding="http://ns.editeur.org/onix/3.0/reference ">
```

This identifies the source encoding of the LIDO output as ONIX for Books 3.0; this is the same as specifying that the namespace for the source elements is "http://ns.editeur.org/onix/3.0/reference". It is worth noting at this point that MINT assigns a new namespace prefix to elements in its input data based on the schema implicit in the instances it has available, and this implied schema therefore does not necessarily include the whole element set of the standard XSD. Note that it was not possible to map all ONIX elements at this time.

8.4.3 Template – lidoRecID

The identifier of the output LIDO record generated by MINT. This is produced by the aggregation process itself and hence lies outside the scope of ONIX for Books. Note that the identifier of the source ONIX record in its original context is not lost but actually captured in the lido:recordID – the lido:lidoRecID is for the *aggregated* record in its new context, linking the original data to its new LIDO expression.

8.4.4 Template – objectPublishedID

The LIDO objectPublishedID is produced from the ONIX ProductIdentifier composite. Exactly parallel to the LIDO record structure, the ProductIdentifier is a composite found directly within the <Product> element, again confirming the compatibility of the basic structure. The mapping uses the XSLT variable map0 to apply ONIX code list 5 (Product identifier type) values to the @lido:type attribute for the objectPublishedID. Note also that multiple public identifiers can refer to the product; there is no single “favoured” identifier, although the mandatory (in ONIX) identifier for the product record, assigned by the record producer (see Appendix 2, section 18.2.1) provides a central ID to link all the public IDs.

8.4.5 Template – category

Similarly to @relatedencoding above, this specifies a category of objects described by the LIDO record, namely, product types as defined in FRBRoo. For the purposes of this mapping, the CRM namespace base URI has been used with the FRBRoo concept code to create a LIDO conceptID, in line with the recommendation that CIDOC-CRM should incorporate the FRBRoo and meta-CRM working drafts into its specification:

```
<lido:category>
      <lido:conceptID lido:type="URI">http://www.cidoc-crm.org/crm-
      concepts/F3</lido:conceptID>
      <lido:term lido:addedSearchTerm="no">F3 Manifestation Product
      Type</lido:term>
</lido:category>
```

This is more general than the @relatedencoding attribute, which, in this case, specifies “products described by ONIX 3.0 records” – however, it is this element that indicates that all LIDO fields are taken to mean type properties rather than individual item properties. For a fuller discussion of this point, see the section on LIDO-CRM mappings in Appendix 2 and section 15.2.5.

8.4.6 Template – [default language of metadata]

The two top-level divisions of the main LIDO record elements are <lido:descriptiveMetadata> and <lido:administrativeMetadata> and their @xml:lang attributes are specified in MINT at the top of the mapping for convenience. They have been set to English for convenience in this mapping work, since the standard sample message is in English, and many publishers will find it convenient to supply ONIX in English.

```
    <lido:descriptiveMetadata xml:lang="en">
    <lido:administrativeMetadata xml:lang="en">
```

ONIX 3.0 does not allow a top-level specification of the product record’s default language, although it recommends this should be agreed between the partners exchanging messages so this LIDO element could perhaps be set manually during the pre-processing stage which in any case will be necessary at the current level of development (see sections 15.2.7 and 15.2.10).

8.5 ELEMENT MAPPINGS – LIDO DESCRIPTIVE

The mappings for the LIDO "descriptive metadata" compose the majority of the mapping as a whole, since the "product information" is the main content of an ONIX for Books message. As will become clear below, a large proportion of the ONIX information content is represented by just one part of the LIDO record; the "classification" structure composed of a concept identifier and corresponding term, both taken from a controlled vocabulary. This highlights the fact that a large amount of "life cycle" information in ONIX messages is, for convenience, conveyed by controled values from the ONIX code lists, even if, semantically, it has much in common with the more granular structures for representing "life histories" in LIDO. The different emphases of the two schemas are apparent here.
Another tension found right across the mappings detailed below is the treatment of "actors"; the directness of LIDO and the indirectness of ONIX. In particular, "actor" entities in LIDO can be explicitly classified by a type with the expected values "person, corporation, family, group". ONIX actors' names tend to be already classified as either person or corporation; otherwise, this distinction is not made explicitly and could only be extrapolated from the types of identifiers used in some cases, or from the context of other elements and values. This reflects the comparatively complex uses envisaged for ONIX data as against LIDO records.

8.5.1 Classification – Object / Work Type

LIDO's Classification wrapper contains only two subsections – "Work Type" and Classification. At a first reading of the LIDO specification it is not obvious how these two parts differ from one another. Both consist syntactically of the same <conceptID> and <term> elements (see the next section, 9.4.2, for a diagram of this structure) so it appears that Work Types are simply another kind of Classification.

Indeed a clarification from one of the LIDO authors[91] confirms that the Work Type is ontologically a sub-class of Classification. As the LIDO specification states, it is "[t]he specific kind of object / work being described"; the LIDO author further restricted this to "the most specific classification that applies to the whole work". This is coherent with the LIDO specification which links Work Type with the SPECTRUM term "Object name", a term used to describe the "collection type" an object belongs to in contexts such as the British Museum[92] or the Getty Research Institute[93] controlled vocabularies.

This definition therefore aligns well with the ONIX class of Product Forms which provide a classification of products within the context of a book retailer's or publisher's "collections". The Product Forms in ONIX code list 150 correspond to the media or format[94] of the product; some examples from list 150 show this (the other entries are more specific but not by many degrees):

Value (code)	Description (label)	Notes (scope note)
AA	Audio	Audio recording – detail unspecified.
AB	Audio cassette	Audio cassette (analogue).
AJ	Downloadable audio file	Audio recording downloadable online.
BA	Book	Book – detail unspecified.
BB	Hardback	Hardback or cased book.
BC	Paperback / softback	Paperback or other softback book.

[91] Stein, R. (2012). Question and answer session on LIDO and MINT at Linked Heritage plenary meeting, Stockholm.
[92] See British Museum Object Names Thesaurus, available at: http://www.collectionslink.org.uk/assets/thesaurus_bmon/Objintro.htm
[93]See CONA, at: http://www.getty.edu/research/tools/vocabularies/cona/about.html
[94] The precise semantics in the ONIX code lists can sometimes mix aspects of the "content" and "carrier" aspects of the product; in any case, both are relevant to what LIDO calls a "work type" since it defines an integral concept of an object's form and function. For more precise distinctions of these concepts the standard references remain the indecs framework and the RDA/ONIX content and carrier analyses.

Value (code)	Description (label)	Notes (scope note)
EA	Digital (delivered electronically)	Digital content delivered electronically (delivery method unspecified).
PI	Sheet music	
PN	Pictures or photographs	
VA	Video	Video – detail unspecified.

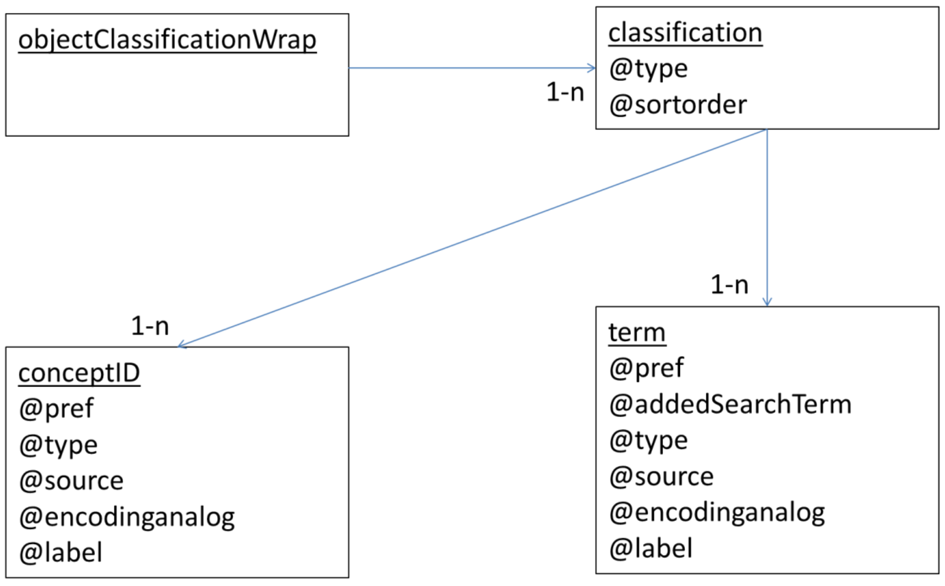
Since <ProductForm> is a mandatory element for ONIX records, and so is <objectWorkType> in LIDO, the constraints of both schemas support this interpretation. This is practical; but the LIDO objectWorkType should also be "the most specific" classification. Product Forms are certainly whole-product classes, but they can be very general. Therefore, from a wide number of possible elements in ONIX, <ProductFormDetail> was also mapped to a LIDO <objectWorkType> to add specificity about the product's medium and format without compromising too far in the direction of classification by one aspect of the product rather than "the whole work". In fact, code list 175, "Product form detail", parallels code list 150 fairly closely in further specifying the formats and media listed there. These two code lists also serve the basic requirement that a retail customer could buy the product found, since the format and media will define in a basic way if the customer will be able to access the product's intellectual content.

Since onix:ProductFormDetail is more specific, it maps to a lido:objectWorkType with a @sortorder="1", whereas onix:ProductForm maps to a lido:objectWorkType with @sortorder="2" so that if both are present, the most specific classification may be preferred for sorting purposes.

One other ONIX element was considered for this mapping <PrimaryContentType> since it is analogous to a "Product Form" for the symbolic/intellectual content of the product. However, since it is primarily intended for ebooks, not mandatory and, in any case, partly inferable from ProductForm values, this was mapped only as a lido:classification.

8.5.2 Classification – Classification

As noted in section 9.5.1 above, the <classification> structure in LIDO has a simple form, pairing a <conceptID> with a <term> in much the same way as ONIX code lists have a value and related description (*e.g.* in list 150 quoted in the section above). This is pictured below in the same kind of simple structure diagram as in section 8.1.1.



LIDO classification structures (simplified hierarchy and cardinalities)

The mapping decision was made to represent ONIX elements with lido:classification if they fulfilled two criteria:

- There is no other more specific class, preferably analogous to their context in ONIX, to map them into using an existing LIDO event or relation structure;
- The ONIX element takes a value from a code list.

This had the positive result of following the LIDO specification which states that a lido:classification "category belongs to a systematic scheme (classification) which groups objects of similar characteristics according to uniform aspects". On the other hand, the ONIX code lists, while sharing the structure of concept ID value and term / label, are not always constructed semantically with this use in mind – certainly they are not always "academic" or "scientific" classification schemes – and they often serve somewhat pragmatic uses rather than conceptually neat descriptions. Thus they often describe very limited aspects of products relevant to particular supply chain partners, or may only be interpretable in context. Semantically this also means that almost all of the @type attributes for lido:conceptID were set to "local" – codes only recognised as part of ONIX messages – although a few widely-used classification standards will be noted later in this section. Finally, this approach also meant that many ONIX elements were mapped to lido:classification even if other related elements (with clearer syntactic-semantic structures) were not.

Syntactically, the ONIX element value (a code list code) was mapped directly to the lido:conceptID and the equivalent code list label mapped to the lido:term using an XSLT variable map (see section 9.2).

A simplified comparison of ONIX elements included in the lido:classification mapping against the example categories of classification given in the LIDO specification shows some of the justification for this decision. Of course many of the ONIX elements used to describe product types correspond only by analogy to the uses envisaged for LIDO, mainly relating to cultural artefacts where manufacture correlates strongly with cultural interest.

LIDO Classification (example categories)	ONIX elements mapped (rough equivalence or analogy)
Material	ProductForm [inferred] ProductFormDetail [inferred] ProductFormFeature(Value)
Form	ProductForm ProductFormDetail ProductComposition
Shape	ProductFormDetail
Function	ProductForm [inferred] ProductFormDetail [inferred]
Region of origin	CountryOfManufacture
Cultural context	CollectionType Language [inferred] AudienceCode AudienceRange Audience ReligiousText EditionType [inferred]
Stylistic period[95]	Language [inferred] PrimaryContentType [inferred] ProductContentType [inferred]

[95] The (highly indirect) inference here would be primarily from the language classification, which can include some historical languages known to be linked to specific historical periods.

LIDO Classification (example categories)	ONIX elements mapped (rough equivalence or analogy)
Museum organisation structure	[none – analogous book trade organisation structure could only be inferred from other fields such as retailer subject headings, publisher-assigned collections *etc.*]
No LIDO equivalent (intellectual content)[96]	PrimaryContentType Language ProductContentType Illustrated EditionType

Some of the most complex syntactic mappings were in this section, because, even though the basic structure (ID, term) is shared by both schemas, ONIX often requires that a value from one list will specify another list to be used for a related element's content. These exceptional mappings are explained briefly in the table below.

ONIX source element	Conditions / correlations
PrimaryContentType	Used to set the Europeana media type by correlating the main groupings of ONIX code list 81 to one of TEXT, IMAGE, SOUND or VIDEO – and if no <PrimaryContentType> element is found, set it to TEXT as a default for book products. The mapping is repeated to give the specific PrimaryContentType value as a separate lido:classification with no conditions.
Language (and subelements)	Condition – the language must be that (or one of those) used (according to the <LanguageRole> subelement) for the text content of the product. Some useful information is lost here as the <LanguageRole> element does not have an analogue in the LIDO classification structure. Also, the various aspects of the text's "language" – language, country variant and script – can be described here but in LIDO their conceptID and term are linked only by their @label.[97]
AudienceRange	The ONIX composite <AudienceRange> uses both the order of its subelements in the XML document, and codelist values to specify the semantics of a sentence of the form FROM EARLIEST-AGE TO LATEST-AGE. This has been mapped to a lido:classification where the part of this sentence is denoted by LIDO's @label attribute. This could alternatively be done using a lido:measurementSet since an age is simply a length of time.
Illustrated	This mapping is the same simple one-to-one correspondence described above, except even further simplified to only use the lido:term with a value of "yes" or "no". The @label attribute indicates this is the answer to the question, "Illustrated?", or "is this product illustrated?". It will be useful for product records that do not contain further details of illustrations (see section 9.5.6).

[96] The absence of any specific classifications for text content or other symbols does not indicate that LIDO cannot express these. The language of transcribed "inscriptions" can be expressed using the xml:lang attribute, for example. However, since LIDO was designed to describe found objects or (not primarily textual) artefacts, it lacks both the most general and the most detailed expressions for classifying text and symbols.

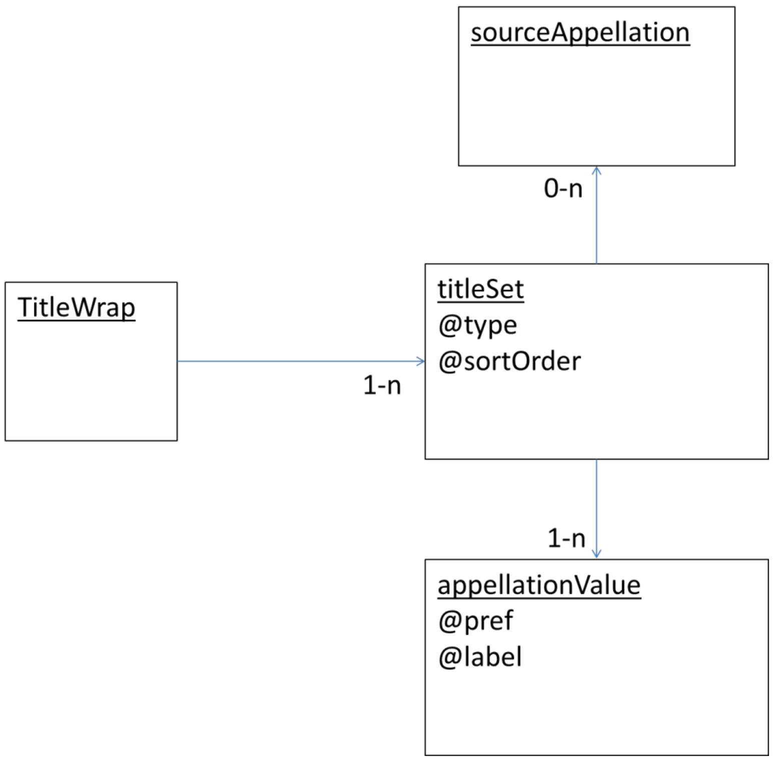
[97] Because, as noted above, the symbolic content of an object or work is somewhat secondary for LIDO, in effect only one "language role" classification is expressed: the product's primary language, since it applies to the whole product.

ONIX source element	Conditions / correlations
ReligiousText	This is another binary flag to show that the product is considered a religious text of some sort. It contains a condition (the <ReligiousText> composite must exist) and maps a single lido:term with value “religious text”.
ProductFormFeature	Uses a large number of conditions based on the <ProductFormFeature**Type**> subelement to select a code list value mapping for the <ProductFormFeature**Value**> subelement which provides the LIDO <conceptID> and <term>. Thus only those Product Form Feature Types which take code list values are mapped as classifications; the others map to descriptive notes (see below, section 9.5.5).

The two ONIX elements Illustrated and ReligiousText are “flags” that convey the simplest possible classification type (one class, to which the product either does or does not belong). Ideally, a URI for the ONIX element itself could be used here to identify membership of this class.

8.5.3 Identification – Title

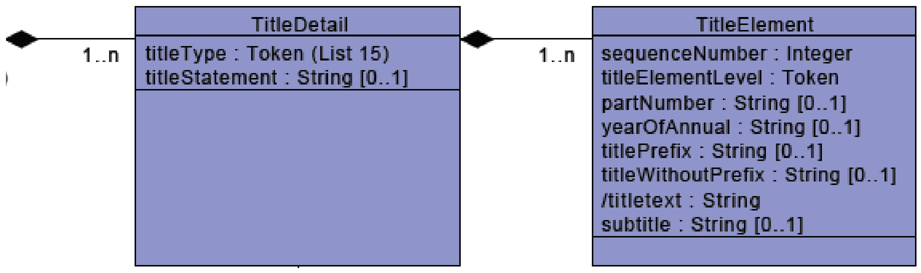
LIDO’s titleWrap syntax only allows for one text string per “title”, with attributes as shown in the simplified hierarchy and cardinality diagram below. This severely restricts what can be usefully mapped to the lido:titleSet construction, especially as the LIDO specification explicitly stipulates “one title or object name and its source information”, which seems to imply that a single, self-contained title per lido:titleSet is expected.



LIDO title structures (simplified hierarchy and cardinalities)

The LIDO specification does supply several useful attributes for the titleSet and appellationValue elements, especially @sortorder and @pref but as shown above, they are only allowed on their respective elements, and @pref is only really a subclass of @sortorder. There are effectively only two levels of detail (“granularity”) as the TitleWrap is really a pure container element with no semantic aspects.

Compare this with the ONIX <TitleDetail> composite structure shown, again in UML, below (noting that not shown is the abstract “title” entity which groups together the actual ONIX XML elements used for this purpose):


TitleDetail
titleType : Token (List 15)
titleStatement : String [0..1]
1..n
TitleElement
sequenceNumber : Integer
titleElementLevel : Token
partNumber : String [0..1]
yearOfAnnual : String [0..1]
titlePrefix : String [0..1]
titleWithoutPrefix : String [0..1]
/titletext : String
subtitle : String [0..1]
1..n

ONIX 3.0.1 title structures (extract from full UML diagram)

Both top elements of this structure are repeatable, and the second contains a third level of detail, compared to LIDO's two levels. The <TitleDetail> contains the semantic qualifier <TitleType> which for this mapping must take the value "distinctive title" *i.e.* the fixed, title proper of the product. The single or multiple <TitleElement> subelements it contains can a sorting order number and a variety of typed structural text-bearing elements for titles or specific parts of titles. The only way to represent these faithfully in the LIDO schema is to either choose those title elements which are already single text strings, or concatenate several subelements of <TitleElement> which are predefined to belong together as one string.

The table on the next page enumerates only the simplest possible combinations of ONIX title *elements* according to the best practice document, and how they have been mapped in the LIDO titleSet. It would be possible to construct algorithms to prefer certain *combinations* of title elements depending which varieties of combinations are present, but there is no indication in the ONIX for Books specification or best practice guides to indicate which are preferred. In any case, MINT does not yet allow such complex conditional statements from the XSLT vocabulary.

Note also that only the very simplest titles and those only where they apply clearly and directly to the product as available for retail, are mapped. No TitleDetails of the type "Undefined" are allowed since it is not clear if they are found on the items in this product class or not, nor if they make up a whole title or only part. Of course other types of title have interest and value for Linked Heritage and Europeana's use case, but the current state of the LIDO schema does not allow them to be represented fully enough for them to be usable either for search or for presentation to end users. Alternative or translated titles could certainly be represented in LIDO using descriptive notes, for example, but this would reduce their semantic precision and also make them less useful for indexing.

	ONIX:								LIDO:	
No.	TitleElement/ TitleElementLevel	TitleElement/ PartNumber	TitleElement/ YearOfAnnual	TitleElement/ TitleText	TitleElement/ TitlePrefix	TitleElement/ TitleWithoutPrefix	TitleElement/ Subtitle	TitleStatement	Map existing elements (not ElementLevel) to simple LIDO titleSet/appellationValue?	titleSet @sortorder
1	-	-	-	-	-	-	-	Y	Y	1
2	01	N	N	Y	N	N	-	N	Y	1
3	01	N	N	N	Y	Y	-	-	Y	1
4	01	N	N	-	-	-	Y	-	Y	2
5	02	-	-	-	-	-	-	-	N	-

Far more combinations are possible in ONIX, but only the first four are possible within the LIDO specification. For all of the first four simplest options, the TitleElementLevel has the value 01, signifying that the title applies directly to *this* product.

- Option 1. is the case where a <TitleStatement> summarises a complex title that cannot be easily constructed by concatenating other ONIX elements. The <TitleStatement> is equivalent to one of the LIDO elements beginning “display” which offer a single text string as an alternative to displaying complex or technical data to the end-user.
- Option 2. represents a single <TitleText> element containing the whole title.
- Option 3. shows the case where the <TitlePrefix> and <TitleWithoutPrefix> should be concatenated.
- In Option 4. there is a <Subtitle> and so this is the only mapping where LIDO’s @sortOrder is set to “2” indicating this titleSet should be displayed after any other. Otherwise, again, the <Subtitle> in ONIX is effectively a single text string.
- All options from 5. onwards have parts taken from the collection and will require either or both of a more complex titleSet in LIDO, and business rules to decide which parts to map in which order.
- Note that the following conditions apply in the XSLT mapping (more complex conditions could be used in future versions):

ONIX source element	Value	Description	Title element combination(s)	Notes
TitleElementLevel	01	Product	All	All titles mapped to LIDO must be "product level" titles as noted above.
TitleType	01	Distinctive title	All	All titles mapped to LIDO must be "distinctive titles" *i.e.* the title which appears on the product, distinguishing it from other related books (*e.g.* other volumes of the same book, other editions). It could be argued that a TitleType of 00 "Undefined" might be acceptable, but this would allow more complex titles that do not fit LIDO's simple title data model.
TitleStatement	[exists]	n/a	1	If there is a TitleStatement, this is the first choice for the LIDO mapping since it is one integral piece of text representing a "title".
PartNumber **YearOfAnnual**	[does not exist]	n/a	2, 3	This condition removes the chance of mapping part of a complex title with part numbers or years.
TitlePrefix **TitleWithoutPrefix**	[does not exist]	n/a	2	This condition removes the chance of mapping a TitleText element that has been mistakenly combined with the use of TitlePrefix and TitleWithoutPrefix.

It would also be possible to concatenate collection-, subcollection- and product-level title elements to create an improvised title string for use in the LIDO appellationValue element. One example used for practical ONIX for Books implementations used the following pattern:

Collection title* (number within collection*) – main title text , part number (year of annual) : subtitle

[* = only for prescribed bibliographic collections, not ascribed collections in ONIX 3.0]

This has so far been ruled out because it would in practice create a new,local <TitleStatement> that is not used by any party in the supply chain, or would necessitate adopting and implementing one of a number of possible title statement standards[98], which would seem to be a task for the LIDO working group;

If a local <TitleStatement> were constructed, the XSLT for this option would be inefficient because MINT presently only allows construction of IF/THEN conditional statements, which means that for every combination of the title elements in the pattern above would require a separate mapping to lido:appellationValue with a set of conditions attached ruling out all the other options, to avoid creating partly empty lido:appellationValue elements with redundant punctuation.

[98] Such as ISBD, Area 1: http://www.ifla.org/en/publications/international-standard-bibliographic-description

8.5.4 Identification – Inscriptions

Here the tension inherent in describing a product type using a schema designed for unique items is very clear. It seems counterintuitive that the text of a printed book or ebook is an "inscription". However, within the definition of the LIDO specification we find : "A transcription or description of any distinguishing or identifying physical lettering, annotations, texts, markings, or labels that are affixed, applied, stamped, written, inscribed, or attached to the object / work, excluding any mark or text inherent in the materials of which it is made." Since the text of a book is not "inherent in the materials of which [the book] is made" it seems clear that we must include "text" printed ("stamped") on the paper pages or "attached" to the ebook file.

The context or use case here is "distinguishing or identifying" a product class (*N.B.* not one "copy of a book" from another since we are exclusively concerned with F3 Manifestation Product Type, not F5 Item). In normal conditions, commercial products should be completely distinguishable by their published identifier (*e.g.* ISBN) and in the worst case, their minimum referent data (title, contributors, publisher, publication date and place, *etc.*). However, since we are viewing the commercial product "as if" it were a heritage object, these conventions cannot be taken for granted (not all of these data will be mandatory in ONIX data either) and in any case, for FRBRoo, the F24 Publication Expression is equally important in identifying the product.

The relevant source element is "ONIXMessage/Product/CollateralDetail/TextContent/Text" and code list 153 categorises the types of content item found there, the relevant codes to select for lido:inscriptions being as follows:

Value	Description	Notes
04	Table of contents	Used for a table of contents sent as a single text field, which may or may not carry structure expressed as XHTML.
05	Flap / cover copy	Descriptive blurb taken from the back cover and/or flaps.[99]
14	Excerpt	A short excerpt from the work.

Note that unlike for Descriptive text (section 8.5.5) and third-party texts (section 8.5.8) no conditions as to audience were necessary here as this is text that appears on the product itself. This is a code list where the optimised nature of the ONIX message becomes clear (very different types of creative content are grouped by their textual nature) and where LIDO's generality becomes useful in coherently aggregating very different types of material.

8.5.5 Identification – Description

The objectDescriptionWrap in LIDO holds textual descriptive notes about the object or work described by the LIDO record, together with optional identifiers and sources for the texts. The LIDO specification suggests that these should be distinct from the object itself (in opposition to lido:inscriptionsWrap) as they should be "a relatively brief essay-like text that describes the entity". In the case of books, this definition could well overlap with that for inscription – in particular, the flap / cover copy identified in the above section 9.5.4. – but since the clear distinction of appearing on the product, or only in the ONIX message is provided, and maps so well to the inscription/description distinction in LIDO, it was decided to use the following criteria for this mapping area:

- The text is in the TextContent composite (as in section 9.5.4.) but does not explicitly appear on the product or in a third-party publication (see section 9.5.8), or
- The text is provided in an ONIX element already described in the ONIX specification as "note", "statement" or "description".

[99] Note that this may change without changing *e.g.* the ISBN of a book product – in cases such as the death of an author or a prize awarded to the book. This does not change the *product* from the publisher's point of view but it does change the *product type* from a heritage point of view (see section 17.1 on "publisher expression") so technically a heritage aggregator should assign a new identifier to this "version" of the product and keep both records.

Note that there is no presumption involved in the LIDO expression that the description text originates from the publisher; if the onix:SourceTitle element is not provided, there will simply be no lido:sourceDescriptiveNote element and thus no statement of the source.

Here are the six mappings together with their conditions or related code lists as for the lido:classification mappings in section 8.5.2.

ONIX source element	Conditions / correlations
IllustrationsNote	Direct mapping of onix:IllustrationsNote to lido:descriptiveNoteValue – on condition that there is no numerical description of illustrations in the ONIX message (in that case, the IllustrationsNote would be mapped within the LIDO measurements structure described in section 9.5.6. below). The XPATH that must not occur in the message is "ONIXMessage/Product/DescriptiveDetail/ NumberOfIllustrations"
ProductFormDescription	Direct mapping to lido:descriptiveNoteValue with no conditions. This is simple textual description of the product's medium and format.
ProductFormFeatureDescription	Direct mapping to lido:descriptiveNoteValue – but with a @lido:type determined by the related <ProductFormFeatureType> element and values mapped from code list 79, which is also used to select only <ProductFormFeatureDescription> elements from composites with codes in list 79 that specify a text description).
EditionStatement	Direct mapping to lido:descriptiveNoteValue with no conditions. This is simple textual description of the product's edition. The rest of the edition elements in ONIX Group P.9 are mapped elsewhere (*e.g.* <EditionType> is in lido:classification).
AncillaryContentDescription	As with <IllustrationsNote> above. The condition for inclusion as a lido:descriptiveNoteValue is that ONIXMessage/Product/ DescriptiveDetail/AncillaryContent/Number does not occur in the message.
TextContent	Complex conditional mapping described below.

The mapping "onix:TextContent/onix:Text" to lido:descriptiveNoteValue depends on the fulfilment of the three conditions below (note that one depends on an attribute of the <Text> element itself):

Subelement of <TextContent>	Allowed values	Descriptions of allowed values	Comments
TextType	02 03 10 11 12 13	annotation Description Promotional headline Feature Biographical note Publisher's notice	The allowed text types are all those either explicitly called a "description" in code list 153 or those types left after excluding "inscriptions" (see section 9.5.4) or clearly linked to a third-party publication event (see section 9.5.8)
ContentAudience	00 03 06	Unrestricted End-customers Students	These three audience categories (from code list 154) are taken to mean "effectively unrestricted".
Text/@TextFormat	00 06 07	ASCII text Default text format Basic ASCII text	The text content must be plain text to allow reuse through LIDO.[100]

[100] Although this ONIX element often contains text marked up in a subset of XHTML (when @onix:textformat="05") this cannot be aggregated in non-display parts of LIDO; nor is it accepted by the

Once these conditions have been fulfilled, there is a nearly perfect semantic correspondence between the relevant ONIX elements and their LIDO targets as shown below:

ONIX	LIDO	Comments
TextContent/TextType	objectDescriptionSet@type	The text type had to be mapped here as there is no @type attribute on the lido:descriptiveNoteValue element, which might be more a specific and thus appropriate place.
TextContent/Text	descriptiveNoteValue	Perfect correspondence – the actual text of the description.
TextContent/SourceTitle	sourceDescriptiveNote	Perfect correspondence – the title of the published source of the text.

8.5.6 Identification – Measurements

The LIDO schema here specifies that measurements must be numerical, expressed in "whole numbers or decimal fractions". Several parts of the ONIX product information can be expressed in this way, including the <Measure> composite used to describe the overall dimensions of printed books and other physical products; the <Extent> composite which describes the length of the product's content[101] in the traditional mode of page count, but also duration (for audiobooks), two composites that provide a count of content items such as pictures or diagrams, and a scale in case of cartographic material. All of these map clearly to the lido:MeasurementsWrap in some way, although ONIX Measure corresponds more closely to the dimensions of the product "as artefact", with <Extent> giving an idea of the size of the symbolic content.

Less obvious is the onix:EditionNumber which refers purely to the process of creating, selecting and otherwise "editing" the intellectual content of the product for final publication. Since this in any case can be adequately represented by an integer number and a type of "count" for the granular LIDO structure (made up of type, unit and value) this element was mapped here too.[102] The related onix:EditionVersion which can contain alphanumeric text, is mapped here in the lido:displayObjectMeasurements element, with a label distinguishing it as a "version" number. Each of these "areas" or explicit composites in ONIX generates a new lido:objectMeasurementsSet so that they are kept separate.

Linked Heritage and Europeana aggregations. It would have to be stripped down to plain text by pre-processing; see section 15.2.5.

[101] "Extent" in ONIX covers 1) page count; 2) duration; 3) file size. Compare with the library cataloguing definition given at: http://www.abc-clio.com/ODLIS/odlis_e.aspx#extentofitem, "the number of physical units comprising the item (example: 356 p. or 13 v.), the specific material designation, and any other details of extent, such as playing time in the case of sound recordings, motion pictures, videorecordings, and DVDs."

[102] Although "edition" could be treated as part of a title because it is normally displayed with the title on products, its nature as a record of the creation and publication processes behind the product makes it more unique, and in any case, ONIX supplies this data independently of titles, possibly since edition details may not always be given in a product's title, and indeed a title may change between editions; see: http://www.abc-clio.com/ODLIS/odlis_e.aspx#edition. Other edition-related elements such as onix:EditionType are mapped differently when not numerical – following the model of illustration and ancilary content numbers and descriptions.

ONIX	LIDO	Value mappings	Comments
AncillaryContent/Number **…AncillaryContentDescription** **…AncillaryContentType**	measurementValue displayObjectMeasurements measurementType	n/a n/a Code list 25	The LIDO container element objectMeasurementsSet (with its subelements) is generated on each occurrence of the onix:Number within AncillaryContent so it should only be mapped when numerical description is supplied.
NumberOfIllustrations **IllustrationsNote**	measurementValue displayObjectMeasurements measurementType="Number of illustrations"		As for onix:AncillaryContent above, the container lido:objectMeasurementsSet is only generated when NumberOfIllustrations occurs in the source.

MapScale	measurementValue
	measurementType="Map scale"
	measurementUnit="1"
			Also mapped to display field with explanatory concatenations.

Measurement **MeasureType** **MeasureUnitCode**	measurementValue measurementType measurementUnit	n/a Code list 48 Code list 50	Perfect correspondence.
ExtentValue **ExtentType** **ExtentUnit**	measurementValue measurementType measurementUnit	n/a Code list 23 Code list 24	Perfect correspondence. Note that when extents are provided in Roman numerals, this is mapped to displayObjectMeasurements as the LIDO schema does specify how numbers should be encoded.

EditionNumber	measurementValue
	measurementType="Edition number"
	measurementUnit="1"
		n/a
		n/a
		n/a
			EditionVersionNumber is mapped to displayObjectMeasurements as it can contain miscellaneous text as well as numbers.

Note that LIDO lacks two aspects of specification that are present in ONIX:

1. Pre-defined list of units to use with measurements;
2. Specification of numeral encoding to use with measurements (*e.g.* Arabic or Roman numerals).

This makes the mapping no less correct but may make aggregated ONIX data less useful in LIDO since it will be difficult to ensure similar measurements appear collated in search results; this will be no less true of heritage data, which may use different units and numeral encodings depending on its source.

8.5.7 Identification – Event (lido:Creation)

Bibliographic records generally hold very limited information directly about the processes involved in creating their objects' intellectual content. Mapping ONIX to LIDO has shown that, at least for commercial product records, this aspect of description is confined to describing the actors in the role of “contributor” in this creation “event”.

Since the lido:event can refer to an extended period of time during which the process takes place, here the definition is taken as broadly as possible to include all types of “contribution”. This highlights the need to acknowledge and reference all of the relevant rightholders, whether for commercial, legal or moral reasons; in contrast to the heritage viewpoint, the various “sub-events” leading up to the fixation of a Manifestation Singleton are not so relevant, as only the final product (normally) appears in a product description.

ONIX	LIDO	Value mappings	Conditions	Comments
	eventType/term="Creation"	n/a		No ONIX element corresponds to this lido:event – it is the implicit context for the Contributor composite.
Contributor **…/sequenceNumber**	eventActor …@sortorder			Each onix:Contributor corresponds to a new eventActor within the same Creation event.
ProfessionalAffiliation/ ProfessionalPosition **…Affiliation**	displayActorInRole	n/a		Concatenation of both subelements to provide "brief biographical information, and roles… of the named actor". LIDO label attribute to distinguish from description below.
ContributorDescription	displayActorInRole	n/a		Simple mapping of descriptive note for actor. Attribute @label to distinguish from above.
NameIdentifer/IDValue **NameIdentifer/NameIDType**	actorID …@pref=preferred …@type	n/a n/a Code list 44		Simple mapping for an identifier for this actor's primary name to the LIDO actor identifier (see fuller discussion of names and actors below this table).
Alternativename/ NameIdentifer/IDValue **AlternativeName/ NameIdentifer/NameIDType**	actorID …@pref=alternate …@type	n/a n/a Code list 44		Simple mapping for ONIX alternative name ID. See discussion below.

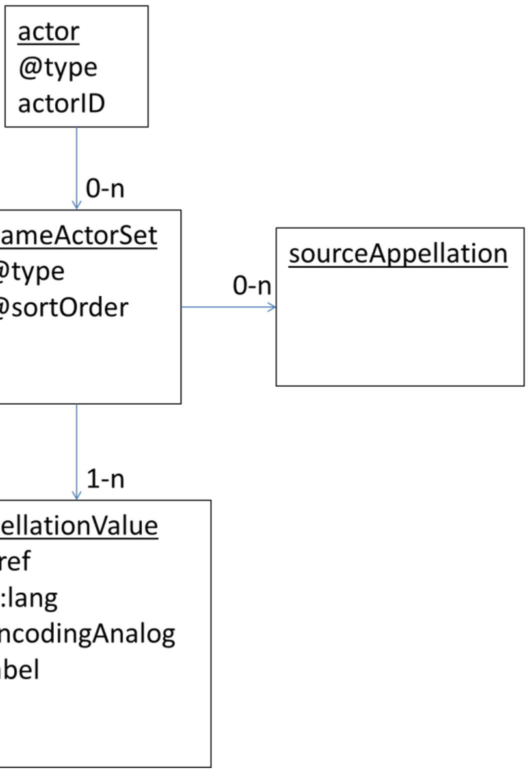
ONIX	LIDO	Value mappings	Conditions	Comments
TitlesBeforeNames *etc.*	nameActorSet appellationValue …@label=”Person name part 1: titles before names” *etc.* @type=”Primary name”	n/a n/a n/a n/a		The mapping for the ONIX Contributor’s primary name. The primacy of this name set is implicit in the ONIX message so there is no structural mapping to lido:nameActorSet.
Alternativename **Alternativename/ TitlesBeforeNames *etc.***	nameActorSet appellationValue …@label=”Person name part 1: titles before names” *etc.* @type=”Alternative name”	n/a n/a n/a n/a		As for the primary name above, but note here the structural mapping to lido:nameActorSet to distinguish alternative names from the primary name and from each other.
ContributorPlace **…/RegionCode** **…/ContributorPlaceRelator** **…/RegionCode** **…/ContributorPlaceRelator** **…/CountryCode** **…/ContributorPlaceRelator** **…/CountryCode** **…/ContributorPlaceRelator**	nationalityActor …conceptID …@type …term …@type …conceptID …@type …term …@type	n/a n/a Code list 151 Code list 49 Code list 151 n/a Code list 151 Code list 91 Code list 151		Simple “classification”-styled pair (see section 9.5.2). The term “nationality” in LIDO is vague, so the more specific ONIX “contributor place” was mapped here with a @type attribute preserving the specific classes of place relationship (*e.g.* born in, died in, worked in…).

ONIX	LIDO	Value mappings	Conditions	Comments

ContributorDate
 	
ContributorDate
	vitalDatesActor
	earliestDate
	…@label=”Date of birth”
	latestDate
	…@label=”Date of death”
		n/a	
			ContributorDateRole=”50”
			ContributorDateRole=”51”
				LIDO contains only “earliest” and “latest” dates, so the conditions here restrict the more expressive ONIX date options to “birth” and “death” respectively. Note the lack of a structural mapping since there is no ONIX container element for multiple onix:ContributorDate sets.

ContributorRole	roleActor/conceptID roleActor/term	n/a Code list 17		This mapping acts exactly like the simple lido:classification mappings in section 9.5.2. and the actor role code list is of course a “local” ID type.

At this point it is useful to compare the structure of names and their relation to the actor entities they are attached to in the data models of ONIX and LIDO. This is not simple, because, as noted in the mapping descriptions above, several parts of the full semantic chains for many elements in both schemas are implicit (in ONIX this is not unusual since it is highly optimised for use in the book industry domain, but for LIDO as a general aggregation schema this could pose problems).

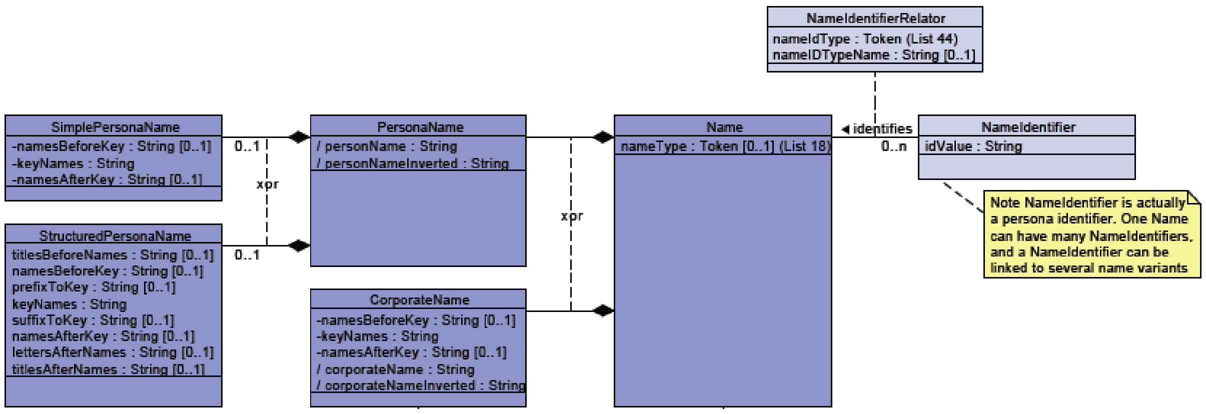


LIDO actor name structures (simplified hierarchy and cardinalities)

The first, simplified schema structure diagram here shows the part of the LIDO <actor> structure that contains names and actor identifiers. The key features to note are that

- The identifier is attached to the lido:actor entity itself, rather than to a name for that actor;
- There are two levels of detail for each "name", and the attributes are shared exclusively by the nameActorSet container and the appellationValue data holder. It is not clear from the LIDO specification why each attribute is reserved to its respective level of description.

This simple structure contrasts with the ONIX names structure pictured in the UML diagram below:


NameIdentifierRelator
nameIdType : Token (List 44)
nameIDTypeName : String [0..1]
SimplePersonaName
-namesBeforeKey : String [0..1]
-keyNames : String
-namesAfterKey : String [0..1]
0..1
xor
StructuredPersonaName
titlesBeforeNames : String [0..1]
namesBeforeKey : String [0..1]
prefixToKey : String [0..1]
keyNames : String
suffixToKey : String [0..1]
namesAfterKey : String [0..1]
lettersAfterNames : String [0..1]
titlesAfterNames : String [0..1]
0..1
PersonaName
/ personName : String
/ personNameInverted : String
CorporateName
-namesBeforeKey : String [0..1]
-keyNames : String
-namesAfterKey : String [0..1]
/ corporateName : String
/ corporateNameInverted : String
xor
Name
nameType : Token [0..1] (List 18)
◀ identifies
0..n
NameIdentifier
idValue : String
Note NameIdentifier is actually a persona identifier. One Name can have many NameIdentifiers, and a NameIdentifier can be linked to several name variants

The model of names in ONIX comes from the experience of assigning identifiers in the commercial world, where one actual person may use several publically available names, perhaps different names in different contexts, and, again in different specific cases, each name maybe be presented in various ways. This leads to at least three discrete levels of identification:

Description level	Explanation	Identification	Description
Person	The actual ("natural") person.	Commercial view:; party to an agreement. Heritage view: attribution and collocation of works; rightholder.	Commercial view: maybe be completely private, restricted information; possibly some contact information shared to allow licensing of work. Heritage view: all information of interest publically shared to enable research and add cultural value.
Persona	A public identity for the person (or occasionally, a group of people).	Contributors are identified at this level.	Commercial view: this is the normal level of identification, with private details linked from the name ID in a separate, restricted database.[103]
Presentation	A textual variant of the name (for example, use of initials or full names; inclusion of name elements; element order).	Presentations of names are not currently identified in either sector.	Usually given as raw text value alternatives for a single name identifier.

For this reason, unique identifiers in ONIX are assigned at the level of the name ("Presentation"), rather than of the actor. This allows alternative presentations to be supplied and linked via "Persona" identifiers such as ISNI, and allows two "Personae" to be linked by supplying identifiers in different name composites of different types[104] *e.g.* "real name" and "pseudonym". Furthermore, bearing in mind the complexity of alternate presentations of names, and the many valid use cases for each, ONIX allows for 8-part structured names, where each part bears a specific relation to the central "key names" used as sorting elements. These relations are based on analysis of actual usage by ONIX data providers, are selected "functionally" to cover requirements for search, sort and display, rather than to express cultural or genealogical construction, and are sufficiently generalised[105] to cover name construction conventions in the most widespread cultures and societies. The elements of the name may be concatenated by the recipient of the ONIX message in different ways depending on the use case: for sorting, indexing or display.

The placement of attributes in the ONIX name again reflects this usage: because a unique identifier can be given for each name, rather than an actual actor/party, the recipient can group together names using a bridging ID like ISNI[106] if they have appropriate access to the data. The order of presentation of contributors in attribution, essential, for example, in academic research papers and core text books, is decided at the contributor rather than name level, and this is reflected in ONIX.

Only two levels of "priority" of names are recognised in ONIX; the primary name (as it appears on the product) and any alternate names; here again LIDO lacks flexibility through assigning the @sortorder attribute to the name element rather than the binary @pref.

[103] See D4.1 section 6.2.1 on the ISNI, designed as a "bridging" identifier between other databases, some of which will probably remain restricted.
[104] See ONIX code list 18, "Person / organization name type".
[105] For example, their definitions only use terms like "name", "suffix", "prefix", "letters" and "title"; specification for use with family, religious or cultural associations, honorifics, linguistic particles, literary or professional status and so on can be defined through examples in the specification and best practice documents, and local guidelines issued by national or language-specific user groups.
[106] See ISNI homepage for more information: http://www.isni.org/

Therefore only a simple mapping of names from ONIX to LIDO was possible, and even this represents a compromise using the broadest possible interpretation of LIDO's specification: "titles, identifying phrases, or names" (lido:appellationValue) is taken to include also *elements of names* so that each part of the ONIX structured name set could be mapped to a separate lido:appellationValue with its own @label containing its element name (all start with "Person name part #", making for easy reconstruction by the LIDO data user). The mapping of the name identifier to a lido:actorID also therefore represents a compromise, although this is really just an ambiguity in LIDO's data model due to its different focus.

Actor roles were mapped simply to the equivalent part of the LIDO entity. It is worth noting that although this mapping is satisfactory, extra semantic richness could be available if a future more detailed mapping (perhaps to an enhanced, revised LIDO) took into account the ONIX <ContentDetail> composite, which identifies and describes text items within a product's content, including their contributors (for example, the case of a book whose chapters each have different authors). Since many of the ONIX controlled values for contributor roles refer to the discrete textual item producted by the contributor (*e.g.* preface, prologue, summary, afterword, notes), and if these contributors could be linked to specifc texts, it might be valuable to consider reflecting the granularity of roles and content items, especially for digital publications.

8.5.8 Identification – Event (from onix:TextContent)

As noted in sections 9.5.4 and 9.5.5, the contents of the onix:TextContent composite are shared between the Inscriptions, Description and Event sections of LIDO. This part of the mapping models the remaining classes of text content as a publication event by a third party (*i.e.* not the book's creators or publishers) including reviews or endorsements of the product. The allowed classes are enumerated below:

Subelement of <TextContent>	Allowed values	Descriptions of allowed values	Comments
TextType	06 07 08 09	Review quote Review: previous edition Review: previous work Endorsement	Note that although types 07 and 08 relate to other products, they were still considered relevant to the product of interest, as relating indirectly to the current work or perhaps its creator(s).
ContentAudience	00 03 06	Unrestricted End-customers Students	These three audience categories (from code list 154) are taken to mean "effectively unrestricted".
Text@TextFormat	00 06 07	ASCII text Default text format Basic ASCII text	The text content must be plain text to allow reuse through LIDO.

Unlike for Inscriptions and Description, this part of onix:TextContent can be modelled as an event as other aspects of the third-party publication event form part of the citation used as "credentials" to back up the text content.

ONIX	LIDO	Value mappings	Comments
TextContent	eventSet	n/a	For the conditions of mapping a TextContent composite here, see above. Each piece of content represents a separate third-party publication event.
	eventType	n/a	The event type was set to "non-specified" although in principle it would be clearer to use eventType="Publication" and a lido:roleInEvent="Subject", if a vocabulary existed for that use.
TextSourceCorporate **TextSourceCorporate**	eventActor …appellationValue	n/a n/a	Similarly to TextAuthor below, this is a simple mapping to one part of the eventActor entity in order to make explicit "name of author of text produced in this event". As above, however, no vocabulary exists to specify the roles involved.
TextAuthor **TextAuthor**	eventActor …appellationValue	n/a n/a	As explained above but with a single person as the author.
ContentDate/Date	earliestDate latestDate	n/a n/a	Condition: ContentDateRole must equal 01, signifying date of publication of this text.
 Text **TextSourceTitle**	eventDescriptionSet …descriptiveNoteValue …sourceDescriptiveNote	n/a n/a	The actual text of the publication and the title of its source. The text is only "a description of the event" in a representative sense.

In summary, this mapping demonstrates the difficulty of creating explicit event entities from implicit events, even when an event is clearly discernible from key aspects of events (dates, actors, products of the event – the text itself). ONIX does not distinguish roles for the actors involved, and LIDO does not (yet) provide vocabularies to designate the object of interest as an input of an event.

8.5.9 Identification – Event (from onix:CitedContent)

The CitedContent mapping is substantially the same as that for TextContent above, with the exception of lido:thingPresent which replaces the eventDescriptionSet noted above. This is because the URL of the CitedContent, which is the essential part of this ONIX composite, and its related descriptions, clearly fit better in the lido:object model.

ONIX	LIDO	Value mappings	Comments
	thingPresent/object	n/a	No more than one piece of text should appear in a CitedContent composite so there is no need for a structural mapping.
ResourceLink	objectWebResource		The Web link element is missing from eventDescriptionSet so this is the only part of lido:event that can take ONIX cited content links.
CitedContentType	objectNote		This and all other mappings below are simple descriptive note mappings, where the @label of the object note denotes the type of description.
SourceType	objectNote		

ONIX	LIDO	Value mappings	Comments
SourceTitle	objectNote		
PositionOnList	objectNote		
ListName	objectNote		
CitationNote	objectNote		

As a third-party publication, this event does not differ substantially from the one for TextContent. It demonstrates some of the differences in describing intellectual content in ONIX and LIDO, especially for Web documents. As with the TextContent publication event, very many enhancements could be imagined but their implementation would depend on a clear use case for data creators.

8.5.10 Identification – Event (from onix:Prize)

The mapping for onix:Prize is an example of a direct semantic correspondence between two schemas where event modeling is followed, as the information conveyed is precisely the relationship of this product to the event of awarding a prize. This event uses most of the core fields of the lido:event structure. It should be noted that, even though, as with the previous two events documented here, the event type is “non-specified”, an award / competition result announcement could conceivably be modelled was a type of “publication” related to the product it is “about”.

ONIX	LIDO	Value mappings	Comments
PrizeCode **PrizeCode**	roleInEvent conceptID term	Code list 41	Even though a passive role, the position awarded represents the product’s “role” in the award event (as opposed to the whole competition, where it was an “entrant”). This is a simple classification mapping.
PrizeJury	eventActor/displayActorInRole	n/a	A simple mapping of a description of the prize awarding body.
PrizeYear	eventDate/earliestDate *etc.*	n/a	Simple mapping of the year of the award to display, earliest and latest date.
PrizeCountry	place/placeID place/placeNameSet/appellationValue	Code list 91	The ISO 3166-1 codes are used here as placeID and to map to country names from the code list.

Because the prize information is already event data, it shows a maximum relational clarity, despite its lack of granularity (owing to its limited importance in the ONIX message). It demonstrates how effectively ONIX data can be mapped into LIDO structures when the semantics are fully explicit.

8.5.11 Identification – Event (lido:Publication)

Together with the Creation event, the Publication event is the main information carrier for a product, which is itself a “publication” as well as a “creation” (*cf*. FRBRoo’s Manifestation Singleton and Publisher Expression types). This part of the ONIX message fits reasonably well within the lido:event framework and yields similar characteristics to the Creation event in its focus on the central Actor information, but adds Place and Date as well since Publication (although the result of a process) can have at least a nominal “point-in-time” date.

ONIX	LIDO	Value mappings	Conditions	Comments
	eventType/term="Publication"	n/a		The ONIX element <PublishingDetail> corresponds to this lido:event but since it is non-repeating there is no need for a structural mapping.
Publisher	eventActor		PublishingRole="01" OR PublishingRole="02"	Identification details of a publisher or co-publisher (respectively) including name, published unique ID and role.
PublisherIdentifier/IDValue	actorID			
PublisherIdentifier/ PublisherIDType	…@type	Code list 44		
PublisherName	actorNameSet/appellationValue			
PublishingRole	roleActor/term	Code list 45		
Imprint	eventActor			The imprint is simply a brand name under which a publisher releases some of their books. Unfortunately when there are multiple publishers and brands in the same message, there is no clear way to correlate an imprint with a publisher unless IDs are provided for both, so they are mapped here as distinct entities.
ImprintIdentifier/IDValue	actorID			
ImprintIdentifier/ImprintIDType	…@type	Code list 44		
ImprintName	actorNameSet/appellationValue			
PublishingDate/Date	eventDate/earliestDate	n/a	PublishingDateRole="01"	The date must be of type "nominal publication date" as all other types are only relevant to supply chain partners.
PublishingDate/Date	eventDate/latestDate	n/a	PublishingDateRole="01"	
	eventPlace			The place ID and name are mapped from the ISO 3166-1 code list just as for PrizeCountry above (section 9.5.10).
CountryOfPublication	…placeID			
CountryOfPublication	placeNameSet/appellationValue	Code list 91		
CityOfPublication	eventPlace/displayPlace			The ONIX city element only contains a name as text and so is mapped to the display element.

8.5.12 Identification – Relation (lido:subjectActor)

The mapping of lido:subjectActor works exactly as the contributor name mapping in section 9.5.7. Therefore the details are not repeated here. The same eight-part ONIX name structure is used in ONIX, and each part is mapped to a single lido:appellationValue in the lido:subjectActor entity.

8.5.13 Identification – Relation (lido:subjectConcept)

A separate lido:subjectSet container is generated for each onix:Subject composite. The lido:subjectConcept contains the usual pairing of conceptID and term found in lido:classification (section 9.5.2) and other controlled vocabulary fields such as role codes. The ONIX concept scheme composite can therefore be expressed almost perfectly, except that the scheme version has no dedicated place. Hence a compromise is reached here by concatenating the scheme version number with the classification scheme name.

ONIX	LIDO	Value mappings	Conditions	Comments
Subject	subjectSet	n/a		
	@sortorder="1"		onix:MainSubject exists	
SubjectCode	subjectConcept/subjectID	n/a		The subject scheme name is mapped from the code list, and concatenated with its version number for the label.
SubjectSchemeIdentifier	…@type	Code list 27		
SubjectSchemeIdentifier	…@label	Code list 27		
SubjectSchemeVersion	…@label			
SubjectHeadingText	subjectConcept/term			
…@language	…@xml:lang			

8.5.14 Identification – Relation (onix:Collection)

This part, and the next three, map ONIX composites that describe related LIDO "works" of some kind – two of these, Collection[107] and ProductPart, are not explicitly "related products" in ONIX – though they can represent abstract sets of products. The other two, Product and Work, have explicit terms defining their relationship to the product of interest.

[107] ONIX 3.0.1 can describe collections of Products using the dedicated <Collection> composite, or indirectly through the <TitleElementLevel> in a <TitleElement> composite, which can indicate a part of a title is inherited from the collection level. Indirect collection description has been omitted from this LIDO mapping, partly for reasons of

Collections in ONIX (typically called “sets” or “series”) are another step of abstraction from product types; they are simply a set of related products, with the relation established by a publisher’s decision or otherwise[108]. Since this mapping proceeds on the assumption that conceptual objects like product types can be mapped to LIDO, there is no reason why these equally abstract objects should not also be. Collections of products are analogous to collections of unique items, which are counted by the LIDO specification as “related objects”.

ONIX	LIDO	Value mappings	Conditions	Comments
TitleStatement	displayObject	n/a		Simple mapping of the single-string title to a display field.
CollectionIdentifier/IDValue **CollectionIdentifier/ CollectionIdentifierType**	objectID …@type	Code list 5		Simple mapping of the identifier.
TitlePrefix **TitleWithoutPrefix**	objectNote			Simple note field concatenation of title elements.
ContributorStatement	objectNote			Simple mapping of the contributor note to the notes field.
CollectionType	objectNote	Code list 148		The collection type refers to the assignment of products to this set by publishers or others.

8.5.15 Identification – Relation (onix:ProductPart)

In contrast to the Collection composite describing collections this product is part of, the onix:ProductPart composite is used to describe parts of this product. Since the mapping is almost identical in form to that in the Collection mapping above, the details are omitted here. The main differences are that no title is provided for the ProductPart, and instead, all of the ProductForm and related elements (see section 9.5.2) are applied to the product part, as well as the <CountryOfManufacture> but these are all mapped to simple descriptive notes, using the same conditions and value mappings as for the lido:classification terms as noted in section 9.5.2.

complexity and limited time, and partly because only a collection title is provided in the ONIX <TitleDetail> so it will produce an incomplete, and mainly implied entity in LIDO, whereas the dedicated <Collection> can provide a full LIDO entity’s content.

[108] ONIX Collections may or may not be products available for retail.

8.5.16 Identification – Relation (onixRelatedProduct)

As for the ProductPart mapping described above, the RelatedProduct composite holds an ID, ProductForm and ProductFormDetail elements, mapped as in 8.5.15. However, it also holds the ProductRelationCode mapped to lido:relatedWorkRelType conceptID and term which is an example of another exact semantic correspondence (though unfortunately there is a lack of structural mapping in MINT for the sets of types). The product relations defined in ONIX code list 51 contains an extremely wide range of relations between products in different forms, and collections that contain them. These are the relator types that can deliver useful semantics for linked data representations of products and the abstract works they manifest.

8.5.17 Identification – Relation (onix:RelatedWork)

The mapping for onix:RelatedWork is even simpler than for RelatedProduct. Only an identifier can be provided (mapped to the LIDO objectID) since there are no ONIX for Books metadata describing abstract creative works directly (although ONIX for ISTC registration does). Of course creative works expressed and fixed as products can be identified by some of the properties of the derived products, so, for example, one of the options for a work identifier is the ID of a product it is expressed and fixed in. The relations that can be predicated of works in ONIX are only five, but again, they have immense potential for creating reliable semantic links between data.

8.6 ELEMENT MAPPINGS – LIDO ADMINISTRATIVE

8.6.1 Rights Work

The LIDO RightsWorkSet contains a highly generalised statement of intellectual property rights in the object of interest. This allows the full onix:CopyrightStatement to be mapped to LIDO, with precise alignment of the entities described.

ONIX	LIDO	Value mappings	Conditions	Comments
CopyrightStatement	rightsWorkSet	n/a/		Structural mapping of ONIX rightsholder statement to equivalent LIDO statement.
	rightsType/term=”Copyright”	n/a		The right type is explicit in the context of the ONIX composite but has to be specified in LIDO.

ONIX	LIDO	Value mappings	Conditions	Comments
CopyrightYear	rightsDate/earliestDate	n/a		
…@dateformat	…@type	Code list 55		
CopyrightYear	rightsDate/latestDate	n/a		
…@dateformat	…@type	Code list 55		
CopyrightOwner	rightsHolder			Structural mapping to create distinct LIDO entities matching multiple ONIX entities.
CopyrightOwnerIdentifier/ IDValue	rightsHolder/legalBodyID			Identifier types are taken from code list 44, so these are typically name or B2B identifiers.
CorporateName	legalBodyName/ appellationValue	n/a		Separate appellationValues generated for either type of name.
PersonName	legalBodyName/ appellationValue			

It is important to note that this LIDO mapping is capable of expressing the entire ONIX copyright information section, and to compare this with the complexity of the ONIX sales rights section (not mapped, see Appendix 4, section 20.4 for details). During the mapping exercise, it became apparent that there is no satisfactory mapping of the ONIX sales rights and markets composites to LIDO due to their combination of elements from actor, event and rights entities.

Sales rights, like licensing terms in PLUS photo metadata, are not declarations of rights held by an existing rightsholder, but rather of rights offered or granted to recipients of the data. This is explicit in schemas like ONIX for Books, but is not envisaged by LIDO and hence cannot yet be expressed.

8.6.2 Record

Both ONIX and LIDO records contain significant amounts of reflexive information about the metadata record itself. However, the ONIX elements corresponding to LIDO's record information are spread across various parts of the ONIX record, illustrating several differences in how the records are created and used. One main difference is that ONIX messages will typically contain multiple product records, and, if the ONIX message originates from a data aggregator, these records will probably originate for one or more sources distinct from the aggregator (in fact, individual data elements may have separate sources, but this is discussed later, in section12.2). This mapping area also contains another Europeana-specific field – rights in the data record – that will need to be addressed in D4.3 because of its central relevance to the licensing framework for data contributors. In addition, this area contains the link to access product retail options, another factor in the commercial case to be outlined in D4.3.

ONIX	LIDO	Conditions	Comments

ONIX	LIDO	Conditions	Comments
Product/RecordReference	recordID …@type="local"		The ONIX message format can contain multiple product records, each with a unique ID (**not** a HTTP URI). This is used to generate the LIDO record ID.
ProductComposition **ProductComposition**	recordType/conceptID recordType/term		LIDO requires a "record type" specifying the unit of description – single item, collection, or group. ONIX contains this information but relates it to the product, rather than the record. This is a minor distinction, since the effect is the same, resulting in a strong semantic correspondence. Note that onix:ProductComposition only denotes singleness or multiplicity – not a specific number of items, which can only be inferred *e.g.* from the onixProductPart composite.
 RecordSourceIdentifier/ IDType **RecordSourceName**	recordSource legalBodyID legalBodyName/appellationValue		There is no structural mapping because a record as such can only have one source in ONIX (although individual elements can have an additional source attribution – see sections 15.2.5 and 15.2.7). This LIDO structure is duplicated for mapping to ESE.
	recordRights (europeana) rightsType/term="CC0"		This field is automatically completed in MINT.
 Header/SentDateTime **RecordSourceIdentifier/ IDValue** **RecordSourceName**	recordRights rightsDate/latestDate rightsHolder/ legalBodyID legalBodyname/ appellationValue		Any rights (*e.g.* copyright) held in the data record by its creator(s). Only the latest date (when the record was transmitted) is known since the date it was created is not provided.
 Header/SentDateTime **Header/Sender/ SenderIdentifier/IDValue** **Header/Sender/ SenderName**	recordRights rightsDate/latestDate rightsHolder/ legalBodyID legalBodyname/ appellationValue		The same as above, but for the rights (*e.g.* database right) of the data sender, perhaps an aggregator of records from many sources.

ONIX	LIDO	Conditions	Comments
Supplier/Website **Supplier/Website/ WebsiteLink**	recordInfoSet recordInfoSet/recordInfoLink	WebsiteRole="40"	The LIDO target specifies a Website for a catalogue entry/data sheet - the ONIX expression is analogous, a customer-facing product site. This instance of the mapping is from a retail supplier.
Publisher/Website **Publisher/Website/ WebsiteLink**	recordInfoSet recordInfoSet/recordInfoLink	WebsiteRole="40"	As above, but a separate instance for the publisher's own Website for this product.

8.6.3 Resource

The LIDO Resource section is a very specific manifestation of the Linked Heritage use case to aggregate textual and visual surrogates of cultural heritage objects. The product data record described in the previous section represents the textual surrogate; the "resource" in this mapping area is intended to refer to a digital image or images of the museum object. For the specific use case in Work Package 4, the visual surrogate could potentially take many forms, from "cover images" of books and other products, to stills taken from part of a film, or a lower-resolution versions of a photo – the only case where the visual surrogate could in theory be a direct mapping from the product itself.

For ONIX data, the most natural choice was the cover image of the book (provided as a URL in the <SupportingResource> composite), which normally appears in product data sheets and retail websites. The ONIX message contains most of the information needed by the LIDO resourceSet as explained below, first for the separate Europeana instance, then for the general mapping to LIDO.

ONIX	LIDO	Conditions	Comments
	resourceSet		For the Europeana instance there should be only link provided, hence there is no structural mapping.

ResourceVersion/ ResourceLink	resourceRepresentation/ linkResource	ResourceMode="03" AND
		ResourceForm="01" AND
		ResourceContentType="01"
			The Europeana requirements are expressed in the conditions: a still image of the book cover, suitable for open publication.

	rightsResource/rightsType/term =" http://www.europeana.eu/ rights/rr-p/"		"Rights reserved" is the only possible option for commercial images.

Within reasonable probability there will only be one link matching the conditions specified above, so no structural mapping to the container element is required to repeat the resourceSet. The Europeana portal image policy[109] does not specify an exact image size for thumbnails and master images, so no condition can be set automatically (this will have to be addressed in D4.3 as it is relevant to the terms for contributing data). No more information than in the two fields above is required (or displayed, currently) by Europeana. In contrast, LIDO can preserve more of the ONIX information provided about the resource, and indeed the entities and structures in both schemas (as would be expected for a primarily technical information set) are almost parallel:

ONIX	LIDO	Conditions	Comments
SupportingResource	resourceSet		The full range of ONIX digital resource entities can be mapped to the equivalent LIDO entity.
ResourceVersion **FeatureValue** **ResourceLink**	resourceRepresentation …@type linkResource	 resourceVersionFeaturetype=”01”	The entity for a version of a digital resource maps exactly. The @type of each representation is mapped to the feature describing file format, with values mapped from Code list 178.

[109] See http://pro.europeana.eu/documents/900548/960640/Europeana+Portal+Image+Policy

ONIX	LIDO	Conditions	Comments
ResourceVersionFeature	resourceMeasurementsSet	resourceVersionFeatureType="02" OR resourceVersionFeatureType="03" OR resourceVersionFeatureType="05" OR resourceVersionFeatureType="06"	Any "features" that are measurable in integers or decimals are selected. The relevant units are selected on the same element.
ResourceVersionFeatureType	measurementType		
FeatureValue	measurementValue	resourceVersionFeatureType="06"	
		resourceVersionFeatureType="05"	
	measurementUnit=""	resourceVersionFeatureType="02" OR	
	measurementUnit="Mb"		
	measurementUnit="pixels"	resourceVersionFeatureType="03"	
ResourceMode	resourceType/term		LIDO resource type is non-repeating, so the most general ONIX classification was used, in line with the LIDO specification. The mapping is a simple ID/term pair.
FeatureValue	resourceDescription	resourceVersionFeatureType="02"	The remaining "feature", the file name, is mapped to a simple descriptive note field.
ContentAudience	resourceDescription		Another descriptive note used to map intended audiences (valid because LIDO allows "contextual" descriptions). Values mapped from Code list 154.
ResourceForm	resourceDescription		ONIX "resource form" details the form of access to the resource. Values from Code list 161.

ONIX	LIDO	Conditions	Comments
ResourceContentType	resourceDescription		ONIX “content type” classifies the subject of the content item, and is thus more specific than could be mapped to lido:resourceType above. Values from Code list 158.

ContentDate/Date
 	
ContentDate/ ContentDateRole
	resourceDateTaken/ date/latestDate
	
	…@label
	(values from Code list 155)
		ContentDateRole=”01”
		ContentDateRole=”04”
		ContentDateRole=”17”
			ONIX content dates for publication, broadcast and last modification are here mapped to a latest estimate of the creation of the resource (thus, no earliest date is mapped).

	rightsResource/ creditLine=” © All rights reserved.”		There is no clear way to extract a rights statement for the resource from the ONIX message. This default value was inserted as a safety measure.

Note from the final part of this mapping that no rights information is carried explicitly for associated digital resources in the ONIX message. This aspect will be covered in detail within D4.3.

8.7 PROGRESS OF ONIX 2.1 MAPPING

Linked Heritage partner, MVB, is currently working to finalise a version of the Excel spreadsheet used to document the ONIX 3.0.1 mapping, replacing ONIX 3.0.1 XPATHs with the relevant XPATHs for the ONIX 2.1 schema, configuring syntactic and semantic mappings to the code list values used in ONIX 2.1 and removing any ONIX for Books elements that were not present in version 2.1.

The current version documents all relevant XPATHs and a complete version is expected before the end of 2012 to accompany this document. Major differences were identified mainly in the <Collection> and <CollateralDetail> blocks.

9 DDEX MAPPING

This mapping is potentially the second most detailed of the four, since DDex is of comparable complexity to ONIX for Books. So far only an outline mapping of DDex instance data to the MINT aggregator has been attempted, but this shows promise. Due to the limitations of the project's time and resources, more attention was devoted to ONIX and the other, more compact schemas in order to fully survey those schemas. It is expected that similar issues to those in the ONIX mapping will also apply to DDex, but with the added aspects of:

- More complex XML structures such as lists of resources that almost certainly will not be mapped within MINT without further XSLT implementation. Because the DDex structure analyses products ("releases") down their component resources and works (it is a less denormalised schema than ONIX), and relates all three types of entity via internal references in the XML document, the XSLT will need to handle many sets of variables and complex conditional statements to rebuild the type of concrete descriptive statements expected by LIDO. This is not yet possible in MINT.
- A larger overall schema with more elements and structures. The data elements from each level of the aforementioned analysis are hierarchically organised for the purpose of allowing rights, licence and deal management by all the partners of the recorded music supply chain, and thus an extra level of "production" detail is present in DDex which is not found in the other schemas (although to some extent in IPTC).

A very general, informal and informative survey of the DDex Release Message is attempted here, to show within the outline of the LIDO schema, where relevant descriptive and administrative data could be mapped, mostly by analogy to the similar ONIX product information message.

LIDO	DDex
`<lido:lido>`	ReleaseID
`<lido:descriptiveMetadata>`	
`<lido:objectClassificationWrap>`	
`<lido:objectWorkType>`	ReleaseType[110]
`<lido:classification>`	Genre, Keywords, CarrierType
`<lido:objectIdentificationWrap>`	
`<lido:titleWrap>`	
`<lido:objectDescriptionWrap>`	GenreText, SubGenre, Synopsis
`<lido:objectMeasurementsWrap>`	NumberOfUnitsPerPhysicalRelease, Duration

[110] For allowed value set, see: http://ddex.net/dd/ERN34-DSR40/DD/ddex_ReleaseType.html

LIDO	DDex
`<lido:eventWrap/>`	Creation event [one per MusicalWork]: MusicalWorkContributor – ID, name(s), role(s) Recording event [one per Soundrecording, but not currently supported by LIDO. "Performance" event type may be suitable]: ResourceContributor – ID, name(s), role(s), ArtistProfilePage IndirectResourceContributor [*e.g.* composers] – ID, name(s), role(s) Publication event [one per Release] OriginalReleaseDate, TerritoryCode, LocationDescription, LabelName
`<lido:objectRelationWrap>`	
`<lido:subjectWrap>`	Character[111]
`<lido:relatedWorksWrap>`	RelatedRelease [one per release] Resource [one for each sound recording released in this product] – ID, Title, SequenceNumber
`</lido:descriptiveMetadata>`	
`<lido:administrativeMetadata>`	
`<lido:rightsWorkWrap/>`	PLine ["phonogram" or sound recording rights statement], CLine [copyright statement]
`<lido:recordWrap/>`	MessageId PartyID [the sender of the message] TradingName [the sender of the message]
`<lido:resourceWrap/>`	DistributionChannelPage [for the retail link to the product in context]
`</lido:administrativeMetadata>`	
`</lido:lido>`	

[111] A narrative protagonist: "A Character is usually an imaginary Party that is represented in a Creation." (see http://ddex.net/dd/ERN34-DSR40/DD/ddex_Character.html)

10 EIDR MAPPING

The EIDR schema is the most compact, since it is referent data set for EIDR itself, the film and television asset identification registry. As part of a DOI implementation, the schema makes full use of the DOI metadata kernel[112] and so can identify objects of many different kinds, including conceptual abstractions, product types, parties involved in creations and users of the registry itself. Only the product types are considered here, although since the registry metadata is optimised for its primary use as a support for interoperability, many entities are described only through identifiers and it may be necessary for any serious use of the data to resolve (some of) these using external datasets. The broader use case notwithstanding, the compatibility of EIDR with LIDO and its relatively small size mean this mapping is relatively simple and reliable.

The only semantically relevant point to note is that the "asset" that is commonly registered in EIDR represents an abstraction one step above the level of the product type. The "encodings" of the audiovisual creation – the distributable versions available for retail – are a secondary entity, since a given AV creation is normally released in a variety of encodings, and one of the key benefits of EIDR is to relate all of these together. Therefore further technical work will be needed for Linked Heritage to ensure that the product can be identified and linked to a retail source.

The syntactic mapping is currently in progress. The table below summarises the semantic mappings identified so far, and some of the relevant conditions, value mappings and other observations, in the same way as the heuristic table for DDex above. Notable absences are any subject classifications, and potential links to retail contexts. These are areas where data would have to be integrated from other sources (and services).

LIDO	EIDR	Comments
<lido:lido>	ID, AlternateID	
<lido:descriptiveMetadata>		
<lido:objectClassificationWrap>		
<lido:objectWorkType>	ReferentType, PackagingClass	The ReferentType classifies the genre of AV creation (*e.g.* movie or TV series), the PackagingClass the type of format (*e.g.* DVD or download).
<lido:classification>	PrimaryLanguage, SecondaryLanguage, Manifestation, StructuralType, Mode, CountryOfOrigin, EditClass, EncodingClass, ColorType, Codec, MPEGProfile, MPEGLevel, PackagingClass	The EIDR "manifestation" here refers to the aspect of the AV creation is in this language (audio, text subtitles). StructuralType and Mode are DOI kernel terms (see footnote 82).
<lido:objectIdentificationWrap>		

[112] For the full description and enumeration of the DOI kernel see: http://www.doi.org/doi_handbook/4_Data_Model.html#4.3.1

LIDO	EIDR	Comments
<lido:titleWrap>	ResourceName, ReplacedAlternateResourceNames, AlternateResourceName, DisplayName	
<lido:objectDescriptionWrap>	Description	Descriptive notes are found in EIDR for specific edits and encodings, as well as audio tracks; these must be short and useful for identification rather than “filmography”.
<lido:objectMeasurementsWrap>	ApproximateLength, Size, BitrateAggregateMass, ApectRation, HeightPixels, WidthPixels, FrameRate	
<lido:eventWrap/>	Creation event: Director, Actor Production event: Publishing event: PrincipleAgent, ReleaseDate, EndDate	The EndDate is used for series in case they have ended.
<lido:objectRelationWrap>		
<lido:relatedWorksWrap>	Clip, Episode, Season, AdjunctContent, AlternateContent	Parts of series and film clips can be described here; also “adjuncts” (supplementary content *e.g.* extras on a DVD) and “alternate” camera angles, audio tracks *etc.*
</lido:descriptiveMetadata>		
<lido:administrativeMetadata>		
<lido:rightsWorkWrap/>	CurrentAssetHolder	
<lido:recordWrap/>	Registrant	
</lido:administrativeMetadata>		
</lido:lido>		

11 IPTC CORE AND EXTENSION MAPPING

The mapping of IPTC “properties” has been completed in its semantic aspect, as well as the LIDO syntax that will be used to represent the full information for an image file. Both aspects are presented in the table below. Note, however, that the mapping as yet cannot be realised in MINT as an XSLT since the XMP files embedded in digital images are not in a format that can be ingested by MINT, and the RDF syntax used with the XMP “wrapper” needs to be reduced to a predictable “schema” for the mapping to be performed in a stable way for all data uploads. A spreadsheet detailing the full semantics, syntax and conditional statements is currently available, with the semantic mappings (detailed below) developed and reviewed in direct cooperation with IPTC. There is agreement with the Europeana Photography project that this agreed mapping should be used for aggregations of IPTC data by Europeana Photography.

Mappings which were particularly successful were those involving the subject of the image, and the use of controlled vocabulary (subject) classifications, as might be expected for the photo industry which lays a strong emphasis on these as tools for image discovery. A common point of interest for commercial and heritage photo curators is the location shown in the image, and this is expressed perfectly using the same semantics and syntax in LIDO as for IPTC, both for the location shown and the location where the image was taken. As noted below, some serious problems arise due to the lack of entity definition across the IPTC syntax and the minimal provision of rights and licensing information in LIDO.

LIDO	IPTC	Comments
`<lido:lido>`	Iptc4xmpExt:DigImageGUID	
`<lido:descriptiveMetadata>`		
`<lido:objectClassificationWrap>`	Iptc4xmpCore: IntellectualGenre Iptc4xmpCore: Scene	
`<lido:objectWorkType>`		
`<lido:classification>`		
`<lido:objectIdentificationWrap>`		
`<lido:titleWrap>`	dc:title	
`<lido:objectDescriptionWrap>`		

LIDO	IPTC	Comments
`<lido:descriptiveNoteValue>`	dc:description Iptc4xmpExt:AddlModelInfo Iptc4xmpExt:ModelAge	Using descriptive notes for model information is unsatisfactory but necessary due to lack of “model” entities in IPTC.
`<lido:repositoryWrap>`		
repositoryName/legalBodyID **workID**	Iptc4xmpExt:RegistryId Iptc4xmpExt:RegItemId	This is the only scheme that explicitly specifies where and as what a product is registered.
repositoryName/legalBodyID **workID**	Iptc4xmpExt:ImageSupplier Iptc4xmpExt:ImageSupplierImageID	A distinct “repository” representing the *e.g.* photo library or agency that supplied *this* copy of the image.
`<lido:objectMeasurementsWrap>` **measurementValue[measurementType=”Height”]** **measurementValue[measurementType=”Width”]** **measurementUnit=”pixels”** **qualifierMeasurements=”maximum available”**	Iptc4xmpExt:MaxAvailHeight Iptc4xmpExt:MaxAvailWidth	Note that some XMP namespaces (*e.g.* exif and tiff) contain measurements for the image at hand, but are not part of the IPTC specification.

LIDO	IPTC	Comments
`<lido:eventWrap/>`		
eventDate/latestDate actorNameSet/appellationValue roleActor/term eventDescriptionSet/descriptiveNoteValue eventPlace (and sub-elements)	photoshop:DateCreated dc:creator photoshop:authorsposition Iptc4xmpExt:DigitalSourceType Iptc4xmpExt:LocationCreated	The eventPlace structure maps the same pattern as the placeSubject structure below, but within the creation event context here.
actorNameSet/appellationValue	photoshop:CaptionWriter	
`<lido:objectRelationWrap>`		
`<lido:relatedWorksWrap>`		
subjectConcept/term subjectConcept/term subjectConcept/conceptID	dc:subject Iptc4xmpCore:SubjectCode Iptc4xmpExt:CVterm	Note that the CVTerm value must be deconcatenated at ":" and split into conceptID and term.

LIDO	IPTC	Comments
placeSubject/place (partOfPlace…) /placeNameSet/appellationValue	Iptc4xmpExt:Sublocation Iptc4xmpExt:City Iptc4xmpExt:ProvinceState Iptc4xmpExt:CountryName Iptc4xmpExtCountryCode Iptc4xmpExt:WorldRegion	These IPTC fields correspond to a hierarchy of locations represented in LIDO by adding “partOfPlace” into the XPATH where noted. CountryCode and CountryName are at the same level.
subjectEvent/event/eventName/appellationValue	Iptc4xmpExt:Event	
subjectActor/actor/nameActorSet/appellationValue	Iptc4xmpExt:PersonInImage	
subjectActor/actor/nameActorSet/appellationValue **subjectActor/actor/actorID**	Iptc4xmpExt:OrganisationInImageName Iptc4xmpExt:OrganisationInImageCode	

LIDO	IPTC	Comments
subjectObject/object/objectNote (and further separate instances with IPTC property name as @lido:label) subjectObject/object/objectID[@type="Inventory number"]	Iptc4xmpExt:AODateCreated Iptc4xmpExt:AOCopyrightNotice Iptc4xmpExt:AOTitle Iptc4xmpExt:AOCreator Iptc4xmpExt:AOSource Iptc4xmpExt:AOSourceInvNo	Almost the full details of the artwork or object portrayed in the image are mapped as objectNotes, which is unsatisfactory. At least a full LIDO description of the object could be linked from the objectID.
</lido:descriptiveMetadata>		
<lido:administrativeMetadata>		
rightsWorkSet/creditLine	dc:rights photoshop:Credit	
rightsHolder/legalBodyName/appellationValue	photoshop:Source	
rightsHolder/legalBodyName/appellationValue rightsHolder/legalBodyID rightsHolder/legalBodyWeblink	plus_1_:LicensorName plus_1_:LicensorID plus_1_:LicensorURL	
rightsHolder/legalBodyName/appellationValue rightsHolder/legalBodyID rightsHolder/legalBodyWeblink	dc:creator plus_1_ImageCreator plus_1_:ImageCreatorID Iptc4xmpCore:CiUrlWork	
<lido:recordWrap/>		

LIDO	IPTC	Comments
<lido:recordID>	xmpMM:InstanceID	
</lido:administrativeMetadata>		
</lido:lido>		

Important properties which could not be mapped due to lack of equivalents in LIDO, or use-specific semantics in IPTC, include:

IPTC property	Comments
Iptc4xmpCore:CiUrlWork	This element holds the URL of a Web site owned by the photo's creator. LIDO cannot express this as part of its eventActor structure, so it has been mapped to a distinct "creator as rightsholder" entity in lido:rightsWork.
dc:creator	This element holding the photographer's name is duplicated in the eventActor and the "creator" rightsHolder in LIDO, in case other optional fields serving this purpose are not included.
plus_1_:MinorModelAgeDisclosure **plus_1_:ModelReleaseID** **plus_1_:ModelReleaseStatus** **plus_1_:PropertyReleaseID** **plus_1_:PropertyReleaseStatus**	These terms from the PLUS rights and licensing vocabulary could find no equivalents in LIDO. They hold important data about the legality of persons and objects portrayed in the image, with specific reference to an agreement or similar documentation.
xmpRights:UsageTerms	Although general rights can be expressed in a fairly granular way in LIDO (see the ONIX copyright statement in section 9.6.1) usage terms cannot.

12 TECHNICAL SPECIFICATION – AGGREGATION PLATFORM

As noted during the evaluation of the ONIX for Books semantic and syntactic mapping (see section 9), many of the problematic parts of a commercial metadata mapping to LIDO arise in part from the semantics and syntax of the LIDO schema itself, and partly from the software implementation of XSLT used to perform the mapping. This section summarises the successes and remaining challenges of using the MINT aggregator and XSLT engine to process commercial sector data.

12.1 DATA UPLOAD AND PRE-PROCESSING

Basic processing of data files at the point of exchange is common to heritage and commercial sector uses. Some issues discovered during the mapping work were common to all four schemas, while others arise from the specific schemas themselves, notably EIDR and IPTC, where there are unusual aspects in the data exchange environment:

- EIDR – data files are actually outputs from the EIDR database, via an API or Web interface, in response to queries from registrants and registered users and hence is normally used internally to other systems connected to EIDR;
- IPTC – semantic properties are embedded in RDF/XML syntax, then an outer layer specifying the XMP file format, and the data is then embedded within the image file itself, for extraction and updating by photo storage, manipulation and archiving software. Although XMP data can be extracted at many stages of the photo workflow, normally it is not separated from the photo content.

12.1.1 Namespaces and schemas

Once data is uploaded to MINT in its current version, a schema is extrapolated from the data instance using the XPATHs actually found therein. A namespace prefix is automatically generated and assigned to this "schema" by MINT and used to identify all source XPATHs in its mappings. This leads to three problems when aggregating commercial data:

1. Not all possible XPATHs may be present in actual sample data (hence the iterative sample data creation method discussed in section 5.5);
2. Sample data from real schema users may contain errors and thus invalid XPATHs that impede the mapping work by creating the illusion that the "schema" imposed by MINT is the standard schema for the data under consideration (hence here the mappings were compared against the standards as in section 5.5);
3. Every new and different data upload generates a new namespace prefix, as it contains new XPATHs and MINT "detects" a new "schema". This is the most serious difficulty for commercial data, since actually the standard schema is usually known and should be used to validate all data ingested by an aggregating system.

These considerations led to the basic step of pre-processing all data uploads, when needed, by hand, so that they specified their own namespace prefix and schema (this was kept constant throughout). It should also be noted that solving this problem will become essential if data is to be aggregated at scale, and for any use in creating linked data representations.

It would be a significant improvement if a standard schema (in the form of an .XSD file) could be uploaded to MINT to be used in creation of a mapping – this would result in the assignment of a standard namespace, and ensure all possible valid XPATHs could be mapped. Later, instance documents should then be checked against the standard schema, and mapped as appropriate.

12.1.2 EIDR

Outputs from the EIDR database contain isolated XML records, from which well-formed XML documents can be created simply by the addition of a root node. Another aspect of pre-processing might be to select only records which describe entities at the product level.

12.1.3 IPTC Core and Extension

This standard results in data which needs the most intense pre-processing. The steps involved will include at least:

1. Extraction of the RDF/XML data from the “XMP packet” (this consists simply of a header identifying the data as XMP so it can be easily removed);
2. Normalisation of the RDF/XML syntax;
3. Extraction of the relevant IPTC Core and Extension properties and assignment of a single, temporary “IPTC Core and Extension” namespace (purely for the aggregation process);
4. Preservation of the various namespaces of the properties, perhaps by transforming them to XML attributes.

The most potentially complex operation will be step 2, since the XMP standard[113] only defines a loose “syntax” for the combination of IPTC’s properties and subproperties (the latter being used mainly for locations and contact details). Thus the RDF/XML elements extracted in step 1 can occur in many syntactical variations, giving rise to alternative XPATHs, each of which would need to be mapped separately to the correct XPATH in the LIDO target. The variations in syntax fortunately do not affect the reference of any properties to the entities identifiable within any XMP file; they are purely convenient structures for software applications to read and write the data, rather than affecting the data model.

Currently MINT does not handle RDF data uploads, which would obviate steps 2. – 4., and hence the mapping cannot yet be implemented in MINT.

12.2 UPDATES

As noted in section 4.5.1, MINT does not link together data records with the same identifier, nor reconcile contradictory records for the same product. In contrast, all four commercial schemas have some kind of (mostly) explicit or (sometimes) implicit support for, or mandate for, updates and de-duplication.

Schema	Elements for updates and deduplication
ONIX	These attributes can be added to all ONIX elements so that individual data items can be prioritised, deduplicated and updated: @datestamp @sourcename @sourcetype Update messages are recognised by the value of onix:NotificationType.
DDex	Two elements can be set to “true” to indicate the current Release Message should replace previous data for the same release: UpdateIndicator – specifies that the message contains updated information; isUpdated – identifies each section which should replace previous data.
EIDR	The registry itself is used to deduplicate data, and specifically identifiers, for assets, and thus is continuously updated.

[113] See http://www.adobe.com/devnet/xmp.html for details of the XMP syntax.

Schema	Elements for updates and deduplication
IPTC	The following XMP properties (not included in the IPTC format) are often used to track versions of the image file as opened, saved and over-written by software applications: xmpMM:DocumentID xmpMM:OriginalDocumentID xmpMM:InstanceID The elements below (sub-elements of the xmpMM:History event structure) specify individual changes made at each step of the file's history: xmpMM:History stEvt:action stEvt:instanceID stEvt:when stEvt:softwareAgent stEvt:changed Finally, the date of the last changes to the XMP metadata set as a whole is held by this element (explicitly endorsed by the IPTC specification): Xmp:MetadataDate

13 METADATA MODEL SELECTION

A substantial part of the research undertaken for this report was the investigation into potential sources of data, both for identifiers and for detailed descriptive, technical and rights metadata. The findings below were drawn from direct correspondence with the experts, standards bodies and other authorities responsible for the protocols, guidelines and licenses (where applicable) governing the use of data and in some cases for the data itself (*i.e.* in the case of ISAN and EIDR, the standards body manages the central registry of all identifiers and supporting data). As noted already in section 2.2.2, the terms of reference for this exercise, from the Linked Heritage Description of Work, were expanded to take into account the wider use cases of providing culturally relevant information and an acceptable and sustainable legal-commercial business case for contributing data – in these cases the findings are necessarily very basic preliminaries to the detailed work on D4.3 to follow them.

Selection criteria	**Books & Audiobooks**		**Film & TV**		**Recorded music & sound**		**Photography**	
	ISBN	**ONIX**	**ISAN**	**EIDR**	**ISRC GRid**	**DDEX**	**PLUS ID**	**IPTC Core and Extension**
Established user base	Estimated 25 million numbers assigned across EU (in 500,000 publisher prefixes) (as of 2011)	User groups in over 20 countries, including many EU states. Several hundred implementers.	240,000 assets registered in EU (c. 37% of 600,000) (as of 2011)	212401 assets registered 183129 have CountryOfOrigin=us (as of 2011)	Estimated 5 million ISRCs, with c. 1/3 in the EU	16 “charter” members including record companies, distributors and authors’ rights representation[114]. Several hundred implementers.	*Strong industry backing but yet to build user base and item registry*	Maintained and promoted by IPTC and CEPIC (both EU)

[114] See http://ddex.net/current-ddex-members-0

Selection criteria	Books & Audiobooks		Film & TV		Recorded music & sound		Photography	
	ISBN	ONIX	ISAN	EIDR	ISRC GRid	DDEX	PLUS ID	IPTC Core and Extension
Adherence to standards and/or standards status in its own right	ISO standard 2108:2005 (ISMN is ISO standard 10957:2009[115] and ISTC is ISO standard 21047:2009[116])	Developed by Assoc. American Publishers; maintained by US and UK book industry groups through EDItEUR	ISO standards 15706-1:2002 and 15706-2:2006 (V-ISAN)[117]	"Promoter" and "contributor" members from all sections of the filmed entertainment supply chain (mainly US but some important EU industry groups) Implementation of DOI, which is ISO standard 26324:2012	ISRC is ISO standard 3901:2001[118] GRid developed and maintained alongside ISRC	Developed and maintained by consortium of record companies, distributors and authors' rights representation	*Proposed industry standard awaiting wide adoption* Supported by leading developers of industry standards	*Standard inherits some ambiguities from Dublin Core* *Standard applied inconsistently*

[115] See http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43173
[116] See http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41603
[117] See http://www.isan.org/pls/portal/URL/PAGE/ISAN_ABOUT_ISAN/ISAN_ABOUT_ISAN/ABOUT_ISAN/70_OFFICIAL_STANDARD/
[118] See http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=59860

Selection criteria	Books & Audiobooks		Film & TV		Recorded music & sound		Photography	
	ISBN	ONIX	ISAN	EIDR	ISRC GRid	DDEX	PLUS ID	IPTC Core and Extension
Demonstrated interoperability with other metadata models, including those familiar to the public sector	Widely used in publishers', retailers', distributors and library's systems; designed to enable supply chain interoperability	Mappings to and from MARC21 already exist; RDA/ONIX framework maps key vocabularies to library terminologies; VMF demonstrates commonalities with CIDOC-CRM	*Lack of known underlying data model; proprietary database output format*	Developed to be interoperable with other standards, including ISAN, ISRC, UPC, Ad-ID[119]	Both widely accepted and used within other standards (*e.g.* DDex)	Based on the same Indecs data model as ONIX so in principle structurally compatible with *e.g.* LIDO; terminological compatibility with library and heritage schemas demonstrable from VMF	Licensing terms data model similar to those used in other domains; PLUS Coalition is partner on the Linked Content Coalition[120] so interoperability expected to be demonstrated by common rights expression model	*Dublin Core basis of some terms could lead to some semantic ambiguity due to IPTC-specific definitions of DC terms;* Organisational openness to interoperability proven by inclusion of *e.g.* VRA and PLUS properties

[119] See http://eidr.org/documents/EIDR_Interoperability_with_Other_Standards_Identifiers_March2011.pdf
[120] See http://www.linkedcontentcoalition.org/Coalition_Partners.html for full list of partners.

Selection criteria	Books & Audiobooks		Film & TV		Recorded music & sound		Photography	
	ISBN	ONIX	ISAN	EIDR	ISRC GRid	DDEX	PLUS ID	IPTC Core and Extension
Demonstrated and/or potential ease of integration with the technologies selected in other thematic work-packages (i.e. Linked Data, PID, selected metadata models)	Stable, unique persistent identifier already in use both with Indecs-based standards and MARC family and CIDOC-CRM data	Based on Indecs so CIDOC-CRM mappable	Stable, persistent identifier *Some ambiguities inherent in standard, mainly the nature of the entity identified*	Hierarchies of entities and relationships, based on the DOI/Indecs data model RDF compatible	Stable, unique persistent identifiers already in use for Indecs-based standards	Based on Indecs so CIDOC-CRM mappable	Stable, unique persistent identifier	*Shares Dublin Core basis with ESE & EDM but uses IPTC-specific definitions of most properties*
Maturity and quality of available technical implementation, documentation and support.	Full documents freely available Strong implementation support	Full documents freely available Strong implementation support	Full documents freely available	Full documents freely available Strong implementation support	Full documents freely available	Full documents freely available Strong implementation support	*Implementation, documentation and support at very early stage*	Full documents freely available

Selection criteria	Books & Audiobooks		Film & TV		Recorded music & sound		Photography	
	ISBN	**ONIX**	**ISAN**	**EIDR**	**ISRC GRid**	**DDEX**	**PLUS ID**	**IPTC Core and Extension**
Technical access to data	Books-in-Print services ISBN agencies and national bibliographies in each EU state if no Books-in-Print exists	Many and various sources, via API or bulk distribution of data from publishers or trade data aggregators	API Mirroring	API Mirroring	Most identifier data probably found in DDex sources; *distributed registries not easily accessible*	Provider-specific data feeds probably available but no standard centralised access likely; *Complex message structure means extra level of data extraction needed*	API	*Most data kept in proprietary databases*

Selection criteria	Books & Audiobooks		Film & TV		Recorded music & sound		Photography	
	ISBN	ONIX	ISAN	EIDR	ISRC GRid	DDEX	PLUS ID	IPTC Core and Extension
Legal access to data	*May be restricted due to catalogue record licensing*	*May depend on many legal and licensing factors, including copyrighted content as part of messages and commercial business models of data providers*	*Data can be freely accessed and re-used for any purpose, including commercial but all metadata must remain associated with the ISAN ID codes; no replication of service*	Available for reuse; *No replication of service*	*May depend on many varied legal and commercial factors due to decentralised nature of registration agencies and existing commercial data aggregators*	*May depend on many legal and licensing factors, commercial business models of data providers, but less use of commercial content within (textual) metadata*	Available for reuse; *No replication of service*	*May depend on many legal and licensing factors, commercial business models of data providers, but less use of commercial content within (textual) metadata*

Selection criteria	Books & Audiobooks		Film & TV		Recorded music & sound		Photography	
	ISBN	ONIX	ISAN	EIDR	ISRC GRid	DDEX	PLUS ID	IPTC Core and Extension
Cost of access to data	May be charged because of catalogue record services; most ISBN data probably found within ONIX data feeds	Data feeds probably available at limited or no cost from some publishers if commercial use case established; *Commercial data aggregators may charge for data supply maintenance*	12000 Swiss Francs for 2 years	Minimum 5000 USD per year	Most identifier data probably found within DDex messages; *Distributed registries, audio archives and data aggregators may have their own pricing models*	Data feeds probably available at limited or no cost from some publishers if commercial use case established; *Commercial data aggregators may charge for data supply maintenance*	Nominal contribution of c. 100 USD per year [likely to change once a registry is available]	Data feeds probably available at limited or no cost from some publishers if commercial use case established

Selection criteria	Books & Audiobooks		Film & TV		Recorded music & sound		Photography	
	ISBN	ONIX	ISAN	EIDR	ISRC GRid	DDEX	PLUS ID	IPTC Core and Extension
Potential to enrich metadata content – schema "articulation", intrinsic marketing content, links to outside sources, shared subject terms, rich description	n/a	High enrichment value due to event-based data model, text content transmitted within messages, links to outside content with supporting data and re-use of shared controlled value lists	*Only minimal identification data available and lack of clear data model or common output format*	*Only minimal identification data available;* Strongly interoperable output format and data model improves likelihood of enrichment from other sources	n/a	High enrichment value due to event-based data model, links to outside content with supporting data; *Little use of common classification vocabularies*	n/a	Rich descriptive content (subjects, classifications, locations, artistic content) with use of shared vocabularies; *Lack of common data model inhibits further enrichment*

Selection criteria	**Books & Audiobooks**		**Film & TV**		**Recorded music & sound**		**Photography**	
	ISBN	**ONIX**	**ISAN**	**EIDR**	**ISRC GRid**	**DDEX**	**PLUS ID**	**IPTC Core and Extension**
Links into existing cultural heritage metadata corpus; common entities (place, actor, event, concept, work) and IDs for them	ISBN already used widely in commercial and cultural heritage data for products; related ISNI and ISTC support future integration of name authority and "work" entity sharing	Support for ISBN confers benefits of existing interoperability with libraries; use of shared concept IDs and cooperative cataloguing should mean some level of shared entities for content (at basic, general level of identification)	Some use for audiovisual archives provides existing links with heritage corpus	Interoperability with ISAN and current work to map EIDR metadata to CEN film archiving standards should enable links in future	Some use in audio archives (*e.g.* broadcast media) may provide some level of shared recording IDs	*Few known existing links beyond possible re-use of ISRCs by audio archives due to business-to-business focus of current implementations*	*No known links with existing heritage data*	Strong links with VRA data in cultural heritage collections; limited use by some photo libraries with cultural heritage images

13.1 OTHER FINDINGS

During the background research and practical mapping exercises for deliverable D4.2, several related areas were investigated which, although not directly contributing to T4.3 or T4.4, are worthy of mention because of their importance to Linked Heritage and the cultural and commercial sectors more generally.

13.1.1 Potential for increased technical interoperability between sectors

The most obvious benefit, beyond the immediate need for Linked Heritage to enable commercial product data to appear in Europeana, is the generalisation of the work on this use case to other areas of collaboration between commercial and heritage sectors. The usefulness of such collaboration has been mentioned already in the discussion on the VMF in section 4.4.6, and based on the premise of Linked Heritage WP4 this usefulness is set to increase.

In particular, in the **photography** sector, there is already a strong overlap in the use cases of photographic libraries, archives and agencies and the heritage organisations which provide subject material for their products. In the context of the 2012 CEPIC Congress in London[121], a group comprising representatives of IPTC, VRA and Linked Heritage, as well as photo library and museum image repository managers identified a need to discuss the provision of heritage information within commercial standards, the mapping of commercial information to cultural schemas, and the use of linked data identifiers within both, as well as the need to express diverse rights information across both sectors.

As noted in section 14, for the **film and TV** sector, the EIDR organisation is investigating interoperability with CEN's film archiving metadata standards, and in future, heritage organisation may wish to register their film assets and use EIDR's flexible metadata model to link together variants and versions, as well as allowing concomitant discovery of commercial products incorporating archival materials, or treating related topics[122].

In the field of **recorded music** it is not clear in which direction(s) cross-sector interoperability will develop, however, it should be noted that the data schema currently in use already provides the semantic and syntactic distinctions needed for extremely detailed cultural data aggregation.

The current move towards linked data models in the **books** world[123] seems likely to build on interoperability with the commercial publishing and book supply industry (which after all supplies many of the objects of interest in the library domain), both in terms of the foundational RDA/ONIX framework already mentioned in section 4.4.6, and the existing MARC/ONIX mappings in use by the library sector (see section 4.4.6) and the on-going work of the RDA developers in IFLA, which explicitly recognise the importance of maintaining compatibility with commercial models[124].

13.1.2 Potential for generating Linked Data

The now ubiquitous "design issues" note outlining how to publish linked data on the Web[125] listed four "rules" or "expectations" for linking data "so that a person or machine can explore the web of data":

- Use URIs as names for things;
- Use HTTP URIs so that people can look up those names;
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL);
- Include links to other URIs. so that they can discover more things.

[121] See section on IPTC Metadata Day at http://www.cepic.org/congress/2012/programme/metadata_iptc_conference and follow-up email group at http://groups.yahoo.com/group/Cultural-Heritage-Metadata/

[122] For an example of a similar business model online, see Ximon at http://info.ximon.nl/en/about-ximon and the discussion of this Web site in D4.1, section 8.2.3.

[123] Especially in libraries –see the Library of Congress' programme of work to find a replacement for the MARC family of formats: http://www.loc.gov/bibliographic-future/

[124] See the "Scope and Principles" section at: http://www.rda-jsc.org/rda.html

[125] Berners-Lee, T. (2010). *Linked Data.* Available at: http://www.w3.org/DesignIssues/LinkedData

The first three rules actually mirror the operation of actionable identifiers already in use in the commercial and library sectors, where a (non-HTTP) URI can be resolved using a service [126] to look up data about its referent, and potentially also other resolvable identifiers. The main difference is to explicitly use the Web architecture itself to represent and deliver the data. However, a less often quoted part of the same document from Tim Berners-Lee recognises the need to define minimal sets of data[127] to attach to any given identifier; this brings up the related need for data models and ontologies specifying *how links will be* made between the identified objects:

- "In practice, when data is stored in two documents, this means that any RDF statements which relate things in the two files must be repeated in each… A set of completely browsable data with links in both directions has to be completely consistent, and that takes coordination, especially if different authors or different programs are involved."
  This approaches the admission that standardised (*i.e.* shared between the producers of the two documents) metadata schemas are still needed.
- "One pattern is to have links of a certain property in a separate document. A person's homepage doesn't list all their publications, but instead puts a link to it a separate document listing them."
  Here we see the need for clearly defined, and shared, semantics at the level of the "classes" of data, amounting to shared semantics, only possible either in very small communities, or those which rely on standardised data models like those described in 4.4.5.
- The design note does not address the issue of "co-reference"[128] directly. It is one of the underlying assumptions of linked data on the Web[129].

Partly in response to the developments mentioned in 13.1.1 above, Bell (2012) emphasises three essential factors for genuine interconnection and enrichment of data:

1. Common public identifiers;
2. Interoperable semantics;
3. Shared vocabularies.

Because these are already in use in the commercial metadata world, and to some extent, thanks to conceptual tools like CIDOC-CRM and data formats like LIDO, in the heritage world too, there are good reasons to think that useful, scalable linked data can be created based on existing technologies, with the right design decisions at the start. Indeed, a recent paper (Tsalapati *et al.,* 2012) on reconciling LIDO with CIDOC-CRM, EDM and linked data formats found several modelling problems which are already at least partly addressed within commercial schemas like ONIX for Books and DDex.

[126] For example, the DOI resolution service: http://www.doi.org/doi_handbook/3_Resolution.html with examples in several fields such as books, journals and scientific data sets (see D4.1, sections 6.3.5 and 6.2).
[127] In harmony with the concept of referent minimum data and scope of identifiers; see for example http://www.doi.org/doi_handbook/4_Data_Model.html#4.3.1
[128] See CIDOC Co-reference Working Group for a discussion of this problem in the heritage field http://network.icom.museum/cidoc/working-groups/co-reference/ and a more technical but practical discussion in the Web context at http://sameas.org/about.php
[129] See the footnote on "context" at http://www.w3.org/2003/04/iri.html#context

13.1.3 Foundations for common standards in rights data communication

Finally, bearing in mind the above points, it is also clear that a new focus on interoperable metadata for rights expressions is needed in both the commercial and heritage sectors. Requirements selected from the commercial sector mappings to LIDO and suggested as optional LIDO extensions are detailed in the next section, although it should be noted that many of these will be the same for heritage uses, especially in the image rights field. The Linked Content Coalition (mentioned in sections 4.5.1 and earlier in section 14) includes one contributor to the IPTC standard (the PLUS Coalition) and will develop a common rights expression model which aims to be interoperable with PLUS standards and Creative Commons licences[130]. The demonstrator project leading on from the initial LCC work will involve Linked Heritage partner NTUA and the MINT platform, so ideally the latter will be developed further to enable commercial metadata management at scale.

[130] See, for example, the June 2012 state-of-play summary: http://www.linkedcontentcoalition.org/uploads/1206120_plenary.pdf

14 CONCLUSIONS

In drawing conclusions from this report it must be acknowledged that the work done here has been experimental in every sense. The semantic mappings created have a high degree of confidence, due to their reliance on the standards and conceptual models identified in D4.1, and the assistance of the relevant experts and standards bodies. However, the domains described by the commercial and heritage schemas are different and the results of D4.2 remain a successful, but not final, first attempt to integrate them. Further work is needed and will be outlined here.

14.1 SUMMARY

Based on the comprehensive LIDO mapping of data in ONIX for Books 3.0.1 product information format, and the tentative mappings of EIDR, DDex and IPTC, Work Group 4 has established that LIDO v1.0 is indeed able to express a practical descriptive record for a wide range[131] of commercial products, for use in the Linked Heritage and Europeana aggregation and display context.

However, this is based on the following *caveats* which go beyond the technical mapping aspect:

- LIDO is used on the assumption that it can be semantically mapped to the FRBRoo modelling concept where all item properties are inherited from the conceptual objects, instances of F3 Manifestation Product Type (including F24 Publication Expression).
- Full, granular descriptive detail of the kind expected by book retail and some library catalogues is not yet achievable (or needed) with LIDO, but could be added without major changes to the LIDO standard (or disruption to its users).
- Expression of rights and licensing terms, beyond the most basic statement of copyright ownership and authors' rights, is not yet possible in LIDO, and must probably be added somehow in any practical use or at scale.
- Data aggregation on a technical level, and on the level of responsible and commercially acceptable maintenance, has not yet been addressed and will require further technical development and the articulation of business requirements and models (at least the latter will be addressed in D4.3).

With this in mind it is now possible to approach possible contributors for fuller datasets, to assess the practical application of the mappings for ONIX 3.0.1, develop complete mappings for ONIX 2.1, EIDR, DDex and IPTC, and discuss possible terms and new services for aggregation of commercial metadata at scale towards Europeana.

14.2 RECOMMENDATIONS

As noted above, and in section 14.1, this report is a only first step towards fuller interoperability across commercial and heritage sectors and has identified significant further tasks for Work Group 4, Linked Heritage, and other interested parties.

14.2.1 For Linked Heritage Work Group 4

The recommendations for Work Group 4 centre on the preparations for D4.3, although they combine tasks of conceptual modelling, technical specification and legal-commercial research. These aspects, as has been stressed, are interrelated; in order to complete task T4.2 Contribution Specification, of course the focus will not be "on technology, but on the legal agreements needed to make this a reality"; however, those agreements will most likely involve some of the technical requirements already familiar from the commercial sector. The main question in the next deliverable will be, how acceptable to commercial providers is the existing level of integration achieved by Linked Heritage, and how much more is to be done. The recommended next steps are thus:

[131] Indeed, products from all four sectors can be represented to some level of detail using only the ONIX for Books schema; the other schemas provide more media-specific detail and are also LIDO-compatible.

1. Continue work on EIDR, DDex and IPTC mappings for review by industry experts and presentation to the project;
2. Liaise with Work Groups 2, 3 and 5 to ascertain feasibility of enhancements of the LIDO schema, further development of the MINT platform, and adoption of controlled vocabularies from the commercial schemas, respectively;
3. Continue to engage commercial and heritage sector experts in dialogue with a view to increased understanding and first-principles interoperability of data models;
4. Begin liaison activities with commercial sector contacts with the aim of
   a. Procuring "test" and "prototype" data sets (see section 15.2.2 below);
   b. Capturing business requirements for large-scale aggregation and data publication (see also 14.2.3 below);
5. Specify legal and licensing problems involved in commercial data aggregation and publishing, and propose possible solutions, in terms of
   a. Presentation of data to the end user (inclusion or exclusion of specific data fields, mappings to ESE and EDM, use of marketing collateral within textual and image metadata, provision of linked content such as text, still image, audio and AV extracts);
   b. Maintenance and management of data;
   c. Cost and sustainability.

It should be noted that substantial desk research towards D4.3 has already been undertaken, so the state of the art in data licensing and re-use is well understood; the main research to follow will now focus on specific case studies and assessing real-world feasibility from the point of view of potential data contributors.

14.2.2 For commercial sector data contributors

Work Group 4 has already addressed the need for clear guidance on the contribution of data sets to the project as part of the Linked Heritage task force that produced a response (published earlier in 2012) to the new Europeana DEA. The guideline can be found in Appendix 7 in full; it makes three main points, clarifying the contribution of:

1. "Test Data" for use in creating and refining mappings from commercial schemas to LIDO;
2. "Prototype Data" for contribution to Linked Heritage and publication to Europeana;
3. Signature of the DEA if (and only if) they provide Prototype Data.

Some commercial companies, including Linked Heritage partner MVB, have already contributed "test data" that has been used in developing the mappings described in this report. Until now, no "prototype data" has yet been published to Europeana. The next step for interested commercial data providers will be to contribute test data if they have not done so already, review the LIDO mapping together with Work Group and discuss their requirements for deriving a subset of elements to publish to Europeana. This subset may be varied according to the agreed legal and commercial framework within which the commercial data is provided.

As noted for the ONIX to LIDO mapping in section 8.4.6, the default language of metadata values (including controlled value sets) is not specified as part of most commercial sector schemas. This should be discussed with Work Group 4, along with any specific controlled vocabularies used by the data provider, to discover if a linked data representation of these values may be possible.

14.2.3 For Linked Heritage

In order to support Work Group 4 in completing its work, the following points are raised for the consideration of the whole project:

1. Prepare sustainability plans and policies as a matter of urgency. One of the most likely and basic concerns for commercial data providers will be the long-term stability and security of their data, of the technical platform (MINT) developed within this project, and their ability to access and manage it if needed.
2. Engage decisively with the wider commercial sector, so that commercial perspectives can be represented to heritage sector partners, common areas of understanding and practical benefits realised, and the efforts of Work Group 4 towards interoperability can be supported. Two key

points of contact will be the Cultural Heritage Metadata interest group of the IPTC (with VRA and PLUS), and the Linked Content Coalition.

Further points are addressed to specific Work Groups within the project.

14.2.4 For Work Group 2

1. Continue to engage, supported by WG4, with the relevant experts in the heritage domain, namely the CIDOC-CRM special interest group, the FRBRoo experts within IFLA and CIDOC's network, and the Linked Heritage partners in their roles as managers of persistent identifiers which are already used both sectors (*e.g.* DOI, ISBN, ISNI) to promote common understanding and initiatives for enhanced interoperability of standards.
2. Assist Work Group 4 in specifying and proposing reasonable enhancements for the LIDO schema to the LIDO working group and explaining their benefits (see point 14.2.5 below).

14.2.5 For the LIDO working group

LIDO was originally developed from museum standards and its focus strongly reflects this. However, as a new standard in its own right, due to the foresight of its designers in basing LIDO conceptually on the CIDOC-CRM and giving it great structural flexibility and modularity, it can be (and has already started to be) used in areas very different from the "unique physical object" environment.

Despite this promising sign of wider applicability, the experiments here suggest that there are still improvements to be made to the schema, not on the level of major redesign (which would be not only unnecessary but also disruptive of existing use) but simply on the level of revision and (optional) extensions to and refinements of specific parts of the schema. These would allow existing LIDO users to continue using the schema, while encouraging new use in two particular areas:

1. Enhanced use in the heritage sector for complex physical objects and conceptual types;
2. Interoperability with commercial sector data as described in this report.

The specific proposals developed during the work of D4.2 are:

1. Complete the mapping of LIDO to CIDOC-CRM taking into account meta-CRM and FRBRoo.
   From a strict conceptual modelling point of view, the mappings reported here all depend on the ability of LIDO to apply its syntax to describe the properties of a class of items, rather than any one specific item. This is the province of what CIDOC's CRM interest group currently calls "meta-CRM", and the domain-specific entities and processes needed for commercial products fall into the library-centred model called FRBRoo. Both meta-CRM and FRBRoo are currently draft extensions to the main (standardised) CIDOC-CRM. In the interests of continuing the Linked Heritage technical work in full harmony with the wider heritage information community, LIDO needs to explicitly acknowledge these models as acceptable mappings for its elements, and detail how their use can be expressed within the LIDO structure (for this, D4.2 has already suggested flagging the record with F3 Manifestation Product Type in the LIDO record category elements).
2. Add granular source element identification.
   The @encodinganalog attribute used to capture "the internal field label of the source database" should be supplemented by an extra, optional attribute for the XML namespace prefix of the "field label" (*e.g.* @prefix and @base URI) so that for specifications such as the IPTC Core and Extension properties (and in fact the other Adobe XMP properties) both the various namespaces and the property names can be separately captured in a structured way. This will also allow interoperability in future with other (linked data?) formats that import two or more namespaces, and support the use of vocabulary management tools such as SKOS and VMF.
3. Implement full support for XML namespaces and URI identifiers for all controlled vocabularies.

ONIX for Books and the other commercial schemas rely on a large number of controlled value lists, some of which seem likely to be expressible in HTTP URI form in the future[132]. This could form a complementary task to the SKOSification work of Work Group 3 (see section 14.2.6 below).

4. Add optional extensions to lido:appellationValue and its containing elements.
   The LIDO elements eventName, legalBodyName, nameActorSet, namePlaceSet and titleSet are used to contain names for events, persons (legal and real) places and objects. However, these elements contain only one sub-element, lido:appellationValue, which represents (whole) "appellations… titles, identifying phrases, or names" and hence is non-repeatable. This does not account for the functional granularity of titles and names typical in bibliographic descriptions, both in commercial and heritage sector data models[133]. Further, these cannot be concatenated reliably due to the large number of possible display options or levels of detail available, and mapping their constituent parts to other sections of LIDO (*e.g.* a treaty signing date mapped from part of a document title into a LIDO event) will tend to make the data less usable and mappings less coherent.
   The recommendation is thus to add two enhancements to these elements:
   a) Attributes expressing relations between title elements and their characteristics should be enabled for all lido:appellationValue elements and their containing (super-)elements:
      @type
      @sortorder
      @pref
      @label
   b) lido:appellationValue should be optionally repeatable and its definition broadened to allow elements of appellations as well as "whole" appellations.

   These two enhancements should enable a maximally complex title (such as those found in ONIX 3.0.1) to be mapped into LIDO. An equivalent alternative would be to introduce a sub-element below the lido:appellationValue (*e.g.* "appellationValuePart") but this might possibly make the revised LIDO incompatible or at least inconvenient for existing users. Making the simple additions recommended here would not change anything in the existing semantics or syntax of LIDO, only add more flexible options, useful to both commercial and heritage data providers. See Appendix 8 for example data using the enhanced schema, also showing how it will retain validation of the existing LIDO XML files. A natural extra enhancement would be to also add an optional appelationID element for each set (or combination) of name elements to align with the "person/persona/presentation" model of names (see section 9.5.7) found in commercial data and identifier systems such as ISNI[134].
5. Enable updating and deduplication at data field level.
   The attributes @sender, @datestamp and @sourcetype should be made available options for all LIDO elements that hold data (as in ONIX for Books 3.0.1). This way the provenance and currency of data can be checked on an element-by-element basis, rather than just per-record. This will harmonise best practice with the commercial sector, allow de-duplication of similar records for the same product, and anticipate some of the likely developments in the use of linked data cataloguing (specifically, the introduction of "quads" rather than "triples").

[132] Indeed some controlled vocabularies in current use already are, like the IPTC newscodes, for example those for "genre": http://cv.iptc.org/Requester?scheme=genre&format=rdf
[133] For example, the MARC format "title statement" (http://www.loc.gov/marc/bibliographic/bd245.html), "personal name" (http://www.loc.gov/marc/bibliographic/bd700.html), "varying form of title" (http://www.loc.gov/marc/bibliographic/bd246.html), "uniform title" (http://www.loc.gov/marc/bibliographic/bdx30.html), and "corporate names" (http://www.loc.gov/marc/bibliographic/bdx10.html) all have detailed structure and optional parts.
[134] See the ISNI F.A.Q. at http://www.isni.org/docs/isni_faq.pdf (second page).

6. Usage terms and conditions for the product (available licences) and related resources.
   Especially for photo products, but perhaps increasingly with other types in future, it is accepted best practice to include details of usage terms and conditions, and other types of available licences, in metadata describing the product. This becomes especially important if the product is on offer for sale (or licence) directly to customers, and for images it is important that use metadata should accompany even preview images. For all products, it is an important element of good customer service practice to only display retail offers to customers who will be able to take them up, distinguishing either by territory, absolute date, time elapsed after some other releases, or a combination. Example structures of licence, sales rights and usage terms and conditions already exist in several commercial schemas[135]; the Linked Content Coalition is developing a highly generalised rights expression data model to interoperate with all of the schemas mentioned here, and several more, including many that will be of relevance to heritage organisations, such as Creative Commons[136].
   As noted in section 8.1.6, a more general rights expression will also certainly be useful for LIDO's cultural heritage users in future as more types of digitised and born-digital cultural work converge across commercial and public sector boundaries.
7. Contact details for licensing.
   Again, as in recommendation 10., the photo sector schema IPTC requires the capacity to add contact details for a licensor and copyright holder to the metadata for an image. Contact details composites appear in other commercial schemas as well (although as yet not in their LIDO mappings for Linked Heritage). To allow present interoperability with photo data, and potential future uses of music and book data, this should be extended both:
   a) in its detail (from simply names, locations and Web sites, to include also telephone numbers, email and postal addresses) and;
   b) context within LIDO (from limited use for "legalBodies" to a detailed, optional information set for any actor within the LIDO schema).

   It may be difficult to model the exact semantic relations of some "contacts" with the object of interest in LIDO. Another problem may be to generalise the concept of "contact details" sufficiently for current and future uses in LIDO (although it is noted that LIDO already allows a highly generalisable representation of "place", and the recommendations here would add a granular "name" structure; links to a number of examples of contact details models and a draft generalised model are found in Appendix 9 as possible starting points for this work.
8. Clarify the mandatory status of ISO 8601 formats for LIDO date elements.
   The specification states: "General format: YYYY[-MM[-DD]]… Format is according to ISO 8601. This may include date and time specification." It is therefore not clear if LIDO strictly intends only YYYY[-MM[-DD]] to be used, or if other ISO 8601 formats are allowed. This is extremely important because commercial schemas, which all of course allow a wide range of date formats to be supplied, may produce instance data with different date types in each file. Some of these may be accepted as LIDO, others not; therefore future mappings must be based on a clear indication. Correct and machine-interpretable dates (such as publication embargo dates) are often critical in commercial applications, and these can only be communicated if date formats are closely specified.
9. Enable specification of numeral encodings and refer to standard lists of measurement units (and languages).
   The lido:objectMeasurementsSet represents measurements in a generic format (unit, value, type) but does not constrain the types of numeral encodings (Arabic, Roman, *etc.*) used for values, nor the units allowed. Both of these could be enhanced by the use of (optional)

[135] For example, in ONIX for Books (see element listing in Appendix 5, Groups P.21, P.24 and P.26.
[136] See for example the most recent LCC project plan:
http://www.linkedcontentcoalition.org/uploads/ProjectPlanv4_April2012.pdf

controlled value lists taken from international standards, if an, optional @lido:type attribute were added to the lido:measurementValue and lido:measurementUnit elements. An optional language and language code scheme attribute pair would also allow direct use of international standard codes rather than the @xml:lang value set, making direct interoperability with more commercial schemas possible.

10. Enable reuse of elements and structures from other areas of LIDO for related work, thingPresent, relatedEvent.
    Several commercial schemas rely on giving extensive details of related products, actors and other entities as part of the relevant information about the product of interest (usually represented by some kind of optimisation, but recognisable as related entities from their structure, context or definition). In LIDO, the related entity structures for events allow the full LIDO entity to be embedded within the top-level event; this should also be allowed at least to some extent, and optionally, for works and actors where they appear as related entities, at least for interoperability with the commercial formats, but also as a step towards providing linked data.
11. Enable full(er) work description for digital resources representing the CHO.
    Part of this recommendation is contained in the above point about enabling more reuse of descriptive structures within related entities in LIDO; the point about allowing usage terms and rights and licensing expressions is also covered here. As mentioned in section 8.3.1, LIDO would benefit from more description of digital photographs as works *per se* so that museums and heritage picture libraries can provide full data both for CHOs and the images that represent them, retaining all information indicated by the best practice for photographs (and increasing interoperability with VRA Core[137]).
12. Allow granular text item descriptions.
    The mappings created to describe third-party publications in the TextContent (section 9.5.8) and CitedContent (9.5.9) event structures, and the (as yet unmapped) ONIX structures allowing granular identification and description of text items within products (see section 9.5.7) point towards the fact that metadata describing any complex, primarily textual product or artefact involves detailed specification of many other texts and their relations to the object of interest at several levels of granularity. With the increase of ebooks and digital humanities scholarship, this need is set to increase. LIDO should attempt to address some of these needs by enabling more (optional) detail for several composites, especially thingPresent and descriptiveNoteSet, as well as eventDescription, to allow these to be used to represent published, Web-referencable text items.
13. Support for structured text items (markup).
    As outlined in the recommendations for WG5 below, MINT does not currently support markup (*e.g.* XHTML or HTML) in metadata imports. The recommendation is for this type of text to be detected and removed, but ideally it could be stored and displayed. In order for LIDO to fully support this use, it should provide technical formatting information about text content.

14.2.6 For Work Group 3

1. Consider the existing published work on VMF in future developments of the Linked Heritage TMP, and consider registering LIDO vocabularies.
2. Include ONIX code lists within terminology mapping experiments, especially
   a. Event types;
   b. Actor roles;
   c. Content and carrier classifications;
   d. Subject classifications;
   e. Product / work relators.

[137] VRA Core is also notably in-scope for CIDOC-CRM: http://www.cidoc-crm.org/scope.html

This has relevance for collaboration with WG2 and WG5 where these make design choices relating to LIDO and MINT, respectively. Creation of linked data representations of LIDO may also benefit from the use of commercial sector vocabularies, even if only as a starting point for more generalised data modelling. The usefulness of these vocabularies to the heritage sector has already been demonstrated by projects such as the VMF and RDA/ONIX frameworks[138].

14.2.7 For Work Group 5

The recommendations for the Technical Integration work group centre on further development of the MINT system to bring it in line with commercial sector best practice for data management[139].

1. Data uploads:
    a. Add pre-processing of XML (and other format) input files.
    As noted in sections 11 and 12, both the EIDR and IPTC data formats require significant pre-processing to render them as well-formed XML files for input to the MINT system and mapping to a new schema. It would be convenient (and essential for large-scale aggregation) to enable a predefined transformation of the input text file (simply adding or removing characters at the start and end of the text) for these formats to make them conform to the XML standard. In the case of the XMP container which is used to transport IPTC properties, a further transformation from RDF/XML to a predictable XML syntax is required; if this can be done using XSLT, as seems likely from the technical investigations in WG4, it would be convenient to apply this as an extra step within MINT's upload manager.
    b. Add pre-processing of HTML markup.
    As noted in section 8.5.5, some ONIX data may include formatting markup for end-user display. For the current use case, XHTML tags should be stripped out to leave plain text. In D4.3 the possibility of including markup to support a sustainable business case will be explored.
2. Schema transformation:
    a. Implement more of the LIDO schema.
    In some parts of the MINT implementation of LIDO, not all allowed elements are available to map. For the complex data schemas found in the commercial world, the full structure of LIDO will be needed (including the enhancements detailed above in section 15.2.5).
    b. Implement more of the XSLT functionality.
    For many commercial schemas, multiple options are available in the data schema which will be selected on the basis of data's providers' individual requirements and the recommendations of best practice guidelines. In order to express these options, XSLT functions such as xsl:choose from within the MINT condition editor, as well as more complex conditional statements, are considerably more efficient than the current workarounds. Also consider including the use of internal variables and IDs, since (in particular in the DDex schema) these may be used in the input data to link together parts of the information record. Finally, some of the existing XSLT code for mapping data values found in XML uploads could possibly be somewhat simplified, and a default value (such as "unexpected") returned when an input value does not match known code lists. At production level, such unexpected values would have to trigger rejection of the output element, and possibly rejection of the record (for mandatory elements).
    c. Add schema validation and XML namespaces.
    Another convenient enhancement would be allowing data providers to upload the XML schema document (XSD) for their data to allow the validation of input files, but also the

[138] Linked Heritage also offers the opportunity for active collaboration with EDItEUR in its role as the developer and maintainer of the code lists themselves; many of those who developed VMF and RDA/ONIX are regular collaborators with EDItEUR.
[139] As NTUA expects to be a partner on the successor project to Linked Content Coalition (Rights Data Integration – see http://www.linkedcontentcoalition.org/uploads/1206120_plenary.pdf, pp30-32) , these would seem timely enhancements, certainly for commercial sector data, and possibly also for heritage use in future.

pre-defined matching of elements to the appropriate namespace. This would be useful for all commercial data, but especially for data formats like IPTC where properties are declared from more than one namespace and would potentially be assigned different prefixes[140]. Most importantly it could be used to map schema-to-schema, as described in section 4.4.4., avoiding the need to create test data that contains all possible XPATHs by creating them from the XSD rather than instance XML data. It could be used to compare schema element definitions directly in the MINT interface by taking them from the XSD documentation where it exists; ideally it could also show semantic relationships between all mapped elements, as found in VMF and possibly in later versions of the Linked Heritage TMP.

3. Database management: (The key here is to ensure that MINT aligns with practices already in use in the commercial world, reducing the burden of maintenance on contributors to Europeana via Linked Heritage).
    a. Updates to data fields.
    It should be possible to upload an updated file, either containing a whole data item or only part of the record, with relevant attributes either for the record item, or for individual data elements, with the result that MINT will produce an updated version including existing data for that record, and replacements made where the datestamp contains a later date.
    b. Party identification for provenance.
    As for temporal updates in the previous point (3.a.), relevant data elements or attributes should be specifiable to enable updates from particular providers to be prioritised in changing the stored record (*e.g.* records from a publisher might be given a higher priority than those from a retailer). This may require checking of attributes on elements and whole records, but also registration of sender IDs, perhaps even at authentication level.
    c. Deduplication (and merge?) of multiple records for a given item.
    Deduplication and merging of data records should be possible at any point, certainly at upload but also when different or duplicate records for a given product are received from different data providers.
4. Data publication:
    Of all the technical recommendations, these will almost certainly be the most critical for the business case to be outlined in Linked Heritage D4.3.
    a. Flexibility of LIDO to ESE mapping or total flexibility in the output (ESE, EDM or “other”) mapping.
    As discussed in detail in sections 3 and 6., any mapping to ESE or EDM (either from LIDO or directly from a commercial standard schema) is really only a question of selection and presentation of elements, rather than a principled decision on semantic and syntactic ground. Therefore the recommendation is to achieve full separation of semantic mapping from presentation to potential end-customers by developing MINT to allow (ideally) complete flexibility in choice of publication schema (*i.e.* incorporate two-stage mapping specifications) or at the very least, allow flexibility in which elements are mapped to the current publication schema, and when.
    b. Inclusion of retail links by merging output data with other sources.
    Even for those data formats where a link to see a product in its retail context can be provided (such as ONIX, with publishers’ and suppliers’ product links), there will very probably be a need to allow other retail offers to be incorporated into an aggregated data set. Whether this is achieved at the data upload, management or publication stage will depend on both technical aspects of the aggregation platform and business requirements of contributors, so it is included here as point to investigate before working on it in depth in D4.3.
    c. Usage terms and conditions for resources, and territorial and temporal (relative and absolute) restrictions on availability of retail products, data about them, and previews of their content.

[140] The IPTC body might in that case need to define a list of namespaces and properties to act as a schema since in practice there is not yet an IPTC or XMP schema defined as an XSD.

Similarly to the previous two points, this is a technical aspect of a commonly accepted business case element. As for retail links in point (4.b.), mechanisms for selectively controlling access to metadata, preview content, and the retail links themselves need to be investigated, either at the level of the initial LIDO aggregation, or via an integrated rights and permissions module added on to MINT.

After discussion of the requirements for data publication above, it should be clear that some of the recommendations for technical enhancements (and aspects of those recommended for LIDO) touch on the role of Europeana as the data publishing end-point; these connections are addressed below in 14.2.9.

14.2.8 For CIDOC-CRM and FRBRoo

Since the work of Linked Heritage partners, both in heritage and commercial organisations, builds on CIDOC-CRM and (in the case of WP4 at least) FRBRoo and metaCRM, and given that many Linked Heritage partners are also active members of one or more CIDOC interest groups, it is appropriate to suggest ways that these experts could develop their work in parallel with ours. This dialogue began as part of the D4.2 work (see for example Appendix 3, a response to questions raised by Work Group 4) and will continue as part of our support for the project and commitment to standards and interoperability.

1. Development and standardisation of meta-CRM and FRBRoo.
   Since the mappings in D4.2 draw their overall validity from the conceptual modelling of meta-CRM (for type attributes) and FRBRoo (for specific media object models), Work Group 4 suggests CIDOC to revisit these related working drafts in the light of the Indecs ontology, the existing Indecs-CRM harmonisation work done in the VMF, and the four specific data standards discussed here, and ideally to produce standard versions of these draft models to enable clearer modelling of commercial products within the heritage data context. In particular, attention could be given to the concept of “object / work type” and its relationship to the RDA/ONIX framework for content and carrier descriptions, as this feeds directly into one of the core mandatory terms in LIDO.
2. Completion of LIDO mapping to CRM and enhancements to LIDO.
   Another aspect of this data modelling convergence would focus on the semantic mapping of LIDO to the CIDOC-CRM, which is required in any case for the modelling of linked data expressions of LIDO. This could centre on key type vocabularies, especially those for actor roles, event types, and media, format and genre classifications, which have already been extensively mapped in the VMF and are a point of contact between the commercial and heritage bibliography communities.
   The use of FRBRoo and metaCRM to map LIDO’s semantics has not so far been considered but we suggest that it should be incorporated to give the maximum flexibility in future to applications of LIDO to data both for commercial products and to other “type” entities in a variety of heritage contexts.

14.2.9 For Europeana

Europeana is the envisaged data publishing end point of key technical requirements of any large-scale commercial data aggregation using these specifications. Work Group 4 will intensify its communication with Europeana, including through the Europeana PPP Task Force, during the work towards D4.3, in order to discuss these and other technical and legal-commercial requirements of commercial data providers.

1. Necessity of updates to Europeana data.
   Since data to be published to Europeana may originate in the commercial sector, all data recipients, including the MINT aggregator, but especially the end point, are expected to allow updates to whole and part records. This may involve more frequent and larger-scale updates than Europeana currently deals with.
2. Support for alternative presentation of data elements.
   As already discussed in the context of MINT (see section 14.2.7), a basic consideration of commercial data publication is the varied presentation requirements of media sectors and even individual data publishers. Therefore WG4 recommends investigating more flexible ways of displaying data in the Europeana portal, and possibly allowing aggregators and other data providers to upload their display options as part of the data publishing process, as well as updating these in the same way as the data set itself.

3. Europeana's core aggregation schema.
As already noted in sections 4.3 and 5, there are serious concerns about the suitability of the (related) ESE and EDM schemas to express existing heritage data, and there are further concerns about their use for commercial data. There will be quality issues with linked data produced from EDM, becoming more serious with each subsequent reuse of such datasets. Therefore EDM linked data may be unsuitable for attracting commercial sector interest in Europeana's linked data publications[141].
Work Group 4 envisages that any commercial data aggregation at scale using the current schemas will treat the Europeana schema mappings primarily in terms of selection and presentation of data elements (which may be uniquely selected based on the data supplier), rather than a semantic transformation. Therefore it is recommended that Europeana, given its commitment to attract data contributions from publishers and other commercial organisations, adopt a more general and extensible schema, either based on an existing standard like LIDO (which after all, unlike Dublin Core, was developed specifically for describing Europeana's objects of interest) or developed from a conceptual basis like CIDOC-CRM, FRBRoo and Indecs (a starting point would be VMF which identifies areas of commonality and complementarity across these models).

14.2.10 For EDItEUR and other commercial standards bodies

There are several small ways in which commercial data schemas could enhance their interoperability with LIDO:

1. Add default language of record(s) element.
The LIDO schema expects a default language for the aggregated metadata at record level. It might be useful, for international commercial use as much as for aggregation, to add similar information for commercial product information. Alternatively, document the use of @xml:lang or best practice on use of an element-level @language attribute.
2. Add expressions for licenses and usage terms for non-commercial "products".
Another small step to assist interoperability might be to investigate explicit support for expressing non-payment or otherwise non-commercial usage terms and licence offers in commercial data. This would go beyond simple statement that a product is free of charge (which is already possible).
3. Create unique URIs for at least the controlled value sets (code lists) used in commercial schemas, and consider a full URI set for the schema elements themselves.
This would enable binary classification mappings such as those in 8.5.2 here, and indeed all types of classification mappings, to be machine-readable and most importantly, updated simulataneously with updates to the code lists and schemas.

14.3 CLOSING EVALUATION

Completion of the ONIX for Books mapping specified in the Description of Work has confirmed many of the initial concerns raised in D4.1 and the literature review for this report. However, it has resulted in concrete and positive recommendations for the Linked Heritage consortium, Europeana and the wider heritage data integration community, to be taken up should they wish to continue the attempt to integrate commercial product information.

Given the complexity of their data models, book and recorded music data seem the least immediately practical targets for integration at the current stage of development; far simpler would be to concentrate on smaller scale aggregations of IPTC and EIDR data since, in both cases, the data model is simpler and more compact, and the need for flexibility in the business model will be met since in any case, agreements would need to be made on a direct basis with the data provider, who is either a central registry in the case of EIDR, or likely to be an aggregator of many smaller providers and thus used to normalising varied data to the IPTC fields.

[141] "In practice, the quality of Linked data implementations is only as good as the data you are linking to, and the meaning and contextualisation of the link you use… [Commercial sector data users will very likely seek out] "curated data", i.e. consistent, managed, linking so you can link to other "quality data" with confidence, while still using the standard Linked Data technologies." See: http://www.doi.org/doi_handbook/5_Applications.html#5.4

With more knowledge of data providers' requirements both for mapping to LIDO and for presentation of data within Europeana, it should be possible to formulate rules for aggregating precisely specified collections of product data for books and music as well; hence the remaining practical work towards D4.3 will focus on examining test and prototype data (see Appendix 5) together with publishers and data providers, to develop a framework for adequately representing these complex product data formats in the Europeana environment.

15 REFERENCES

15.1 CITED IN THE REPORT

Adler, M. J. (1983). *How to speak, how to listen.* New York: Macmillan.

Adler, M. J., & Van, D. C. L. (1972). *How to read a book.* New York: Simon and Schuster.

Available at: http://webdoc.sub.gwdg.de/edoc/aw/d-lib/dlib/january99/bearman/01bearman.html

Bearman, D. *et al.* (1999). *Progress report on reconciling metadata requirements from the Dublin Core and INDECS/DOI Communities.* D-Lib Magazine. 5(1).

Delmas-Glass, E. (2012). Using open source tools to expose cross-collection data in the LIDO schema; Part I: LIDO at the Yale Center for British Art: modeling collection data for linked open data. *CIDOC Conference 2012 – Enriching Cultural Heritage, Helsinki, Finland.* Available at: http://www.cidoc2012.fi/en/File/1607/delmas.pdf

Doerr, M. (2010). Technological Choices of the ResearchSpace Project. [online]. Available at: http://www.researchspace.org/researchspace-concepts/technological-choices-of-the-researchspace-project [accessed December 2011].

Godby, C. J. (2012). *A crosswalk from ONIX version 3.0 for books to MARC21.* Available at http://www.oclc.org/research/publications/library/2012/2012-04.pdf

Godby, C., Smith, D., and Childress, E. (2003). *Two Paths to Interoperable Metadata*. Paper presented at the 2003 Dublin Core Conference, DC-2003: Supporting Communities of Discourse and Practice—Metadata Research & Applications, September 28-October 2, in Seattle, Washington (USA). Available online at http://www.oclc.org/research/publications/archive/2003/godby-dc2003.pdf

Godby, Carol Jean. 2010. *Mapping ONIX to MARC. Report and crosswalk produced by OCLC Research.* Available online at: http://www.oclc.org/research/publications/library/2010/2010-14.pdf (report) and http://www.oclc.org/research/publications/library/2010/2010-14a.xls (crosswalk).

Hopwood, M. (2012). *Best Practice Report – Public Private Partnership.* [Linked Heritage Deliverable D4.1] Available at: http://www.linkedheritage.org/getFile.php?id=283

i2010: Digital Libraries High Level Expert Group – Copyright Subgroup. (2008). *Final Report on Digital Preservation, Orphan Works, and Out-of-Print Works.* European Commission. Available at: http://ec.europa.eu/information_society/activities/digital_libraries/doc/hleg/reports/copyright/copyright_subgroup_final_report_26508-clean171.pdf

ISO/DIS 25964-2, *Information and documentation — Thesauri and interoperability with other vocabularies — Part 2: Interoperability with other vocabularies*

Magellan Media Consulting. (2012). *The development, use and modification of book product metadata.* Available from http://www.bisg.org/publications/product.php?p=27

Miriam, J., & McGlinn, M. (2002). *The trivium: The liberal arts of logic, grammar, and rhetoric : understanding the nature and function of language.* Philadelphia, PA: Paul Dry Books.

Nielsen Book Data. (2012). *The Link Between Metadata and Sales.* Available at http://www.nielsenbookdata.co.uk/uploads/3971_Nielsen_Metadata_white_paper_A4(1).pdf

Stein, R. *et al.* (2005). Das CIDOC Conceptual Reference Model: Eine Hilfe für den Datenaustausch? *Mitteilungen und Berichte aus dem Institut für Museumskunde.* SMB Staatliche Museen zu Berlin.

Tsalapati, E., Simou, N., Drosopoulos, N. and Stein, R. (2012). Evolving LIDO based aggregations into Linked Data. *CIDOC Conference, 2012 Helsinki.* Available at http://www.cidoc2012.fi/en/File/1663/simou.pdf van den Heuvel, C., & Rayward, W.. (2011). Facing interfaces: Paul Otlet's visualizations of data integration. *Journal of the American Society for Information Science and Technology*., 62(12)., p2313. (Document ID: 2511361761).

Weston, A. (2000). *A rulebook for arguments.* Indianapolis: Hackett Pub. Co.

15.2 INDICATED READING LIST

Bountouri, L., Papatheodorou, C., Soulikias, V., & Stratis, M.. (2009). Metadata interoperability in public sector information. *Journal of Information Science*, 35(2), 204.

Diego Calvanese, D. & Giuseppe De Giacomo, G. (2005). Data Integration: A Logic-Based Perspective. *AI Magazine*, 26(1), 59-70.

Campbell, D Grant, & Fast, Karl. (2001). The ontological perspectives of the Semantic Web and metadata harvesting protocol: applications of metadata for improving Web search. *Canadian Journal of Information and Library Science*, 26(4), 5-19.

Benjamin Carter, B. (2000). XML: Filling a data-description gap, part II. *Journal of Database Management*, 11(2), 30-33.

Chung, C., *et al.*(1999). Knowledge and object-oriented approach for interoperability of heterogenous information management systems. *Journal of Database Management*, 10(3), 13-25. (Document ID: 41798832).

Ding, H. (2005). Integrating semantic metadata in P2P-based digital libraries. *Library Management*, 26(4/5), 218-229.

Hao Ding, H. & Ingeborg Sølvberg, I. (2007). Rule-based metadata interoperation in heterogeneous digital libraries. *The Electronic Library*, 25(2), 193-206. (Document ID: 1433332231).

Ding, Y., *Jacob, E., Fried, M., Toma, I., Yan, E., Foo, S., & Milojevic, S.. et al.* (2010). Upper tag ontology for integrating social tagging data. *Journal of the American Society for Information Science and Technology*, 61(3), 505.

Doerr, M. (2000).*Mapping of the Dublin Core Element Set to the CIDOC CRM*: *Technical Report 274.* ICS-FORTH. Available at: http://www.cidoc-crm.org/docs/dc_to_crm_mapping.pdf

Dongwon, Jeong., *Peter Hoh In, Fran Jarnjak, Young-Gab Kim, & Doo-Kwon Baik. et al.* (2005). A message conversion system, XML-based metadata semantics description language and metadata repository. *Journal of Information Science*, 31(5), 394-406.

Gordon Dunsire, G. & Mirna Willer, M. (2011, March). Standard library metadata models and structures for the Semantic Web. *Library Hi Tech News*, 28(3), 1-12.

Eulalia Roel. (2005). The MOSC Project: Using the OAI-PMH to Bridge Metadata Cultural Differences across Museums, Archives, and Libraries. *Information Technology and Libraries*, 24(1), 22-24.

Joanne Evans, J., Barbara Reed, B. & Sue McKemmish, S. (2008). Interoperable data :Sustainable frameworks for creating and managing recordkeeping metadata. *Records Management Journal*, 18(2), 115-129.

Hill, Linda L., *Janee, Greg, Dolin, Ron, Frew, James, & Larsgaard, Mary. et al.* (1999). Collection metadata solutions for digital library applications. *Journal of the American Society for Information Science*, 50(13), 1169-1181.

Senator Jeong, S. and Hong-Gee Kim, H. SEDE: An ontology for scholarly event description. *Journal of Information Science* 2010 36: 209 originally published online 5 February 2010. DOI: 10.1177/0165551509358487

Lee, S., & Jacob, E.. (2011). An Integrated Approach to Metadata Interoperability. *Library Resources & Technical Services*, 55(1), 17-32.

R. William Maule, R. (2011). "Cognitive maps, AI agents and personalized virtual environments in Internet learning experiences. " *Internet Research* 8.4 (1998): 347. Web. 8 Dec. 2011.

Norm Medeiros, N. (2006). Metadata in a global world. *OCLC Systems and Services*, 22(2), 89-91.

Ron Miller, R. (2003, March). Get it together: Integrating data with XML. *EContent*, 26(3), 20-24.

Park, J., & Tosaka, Y.. (2010). Metadata Creation Practices in Digital Repositories and Collections: Schemata, Selection Criteria, and Interoperability. *Information Technology and Libraries*, 29(3), 104-116.

Pentaris, F., & Ioannidis, Y.. (2004). Mapping objects. *International Journal on Digital Libraries*, 4(1), 52-55.

Raths, D. (2008, April). Sharing data in a crisis-State and local groups work on interoperability. *KM World*, 17(4), 16-17,29..

16 APPENDIX 1 – GLOSSARY OF TERMS

Term	Synonyms	Definition (illustrative)
Attribute	Property [but see special use in 2.]	1. A characteristic of an entity; 2. XML: a data container attached to an element, with no sub-containers.
CHO [cultural heritage object]	Artefact, [art] work, [museum] object	The "provided" object of interest for Europeana and Linked Heritage, described by a metadata record.
Class	Type	A set of entities completely defined by a list of shared properties.
Collection	Series, set	A set of entities, not necessarily sharing any properties other than membership of the collection as defined by some authority such as a heritage institution or publisher.
Composite	Data structure	In ONIX: A structured grouping of data elements in an XML data file or schema. The LIDO schema defines contains composites normally with the suffix –Set or –Wrap in their container element name.
Container [element]	Structural element [LIDO]	An XML element in a schema used only to hold other elements.
Controlled value list / controlled vocabulary / terminology	Code list [ONIX]	A list of terms and their meanings, usually also represented by codes for brevity, which is maintained and developed, often publically for shared use (like ONIX code lists, BIC and BISAC subject headings) but sometimes internally to a company. In contrast to an XML schema's element set, no formal data structure is defined by a controlled value list, but some logical relationships may be defined between (some of) the terms.
Data element		An XML element that directly contains data in the form of text characters.
DO [digital object]	Digital surrogate, digital image, resource	A digital image file representing the CHO in the Europeana and Linked Heritage context.
Entity [with reference to a specific data schema]	Resource	1. Broadly, anything referred to by an identifier; 2. In schema mappings: a coherent, related set of structural and data elements within a schema (["an ONIX entity", "LIDO entities, *etc.*)], which potentially could form the object of description in another data record.
Expression		1. Information structured and defined by a particular data schema ("an ONIX expression"); 2. FRBR and FRBRoo: a particular realisation or version of a (purely) conceptual creative work; 3. indecs (*i.e.* ONIX, EIDR, DDex): a perceivable creative work.
Instance [data]		1. Generally, a more concrete example of an abstract class; 2. XML: a data file created according to a given schema.

Term	Synonyms	Definition (illustrative)
Life history		Historical events relating to a CHO such that the CHO can be considered somehow a document or witness to their occurrence and interpretation.
Lifecycle	Workflow	Creative and industrial events resulting in the production of a creative work.
Linked open data	Semantic Web	Publication of raw data on the World Wide Web, using HTTP URIs to identify entities and providing links between them as well as other data about them; an "open" licence for reuse of the data was considered part of this method from its original conception.
Manifestation		A set of fixed items of a particular medium, format *etc.* that make accessible a creative work. This has the same meaning in FRBRoo and indecs data models.
Mapping	Crosswalk	Matching syntactic, semantic and possibly technical aspects of two or more data models directly to each other.
Namespace		XML: An identifier for a set of terms, linking their names to a particular set of definitions to remove ambiguity.
Natural language		Normal human language, written or spoken conversation or discourse.
Normalise		1. Databases: organizing the fields and tables of a relational database to ensure each piece of data is stored only once, and that there are no internal inconsistencies in the data. 2. Aggregation: re-expressing many disparate data sets in a standard format (*e.g.* LIDO).
Object [of interest]	Resource; CHO; product [type]	The stable, coherent entity described by a metadata record. Can be perceivable (like a CHO) or conceptual (like a product type).
Person/persona/ presentation [name models]	Name	Terms used to differentiate 1) a natural person having one or more names; 2) their "names" – public identities; 3) textual variants of each name.
Primitive semantics	Definitions;	The definition of a term in natural language, not decomposed further for the purposes of the particular use.
Product	Manifestation	The set of functionally identical items that constitutes a manifestation of creative expressions; examples are publications, music and audiovisual releases; photographic images.
Semantics	Definitions; denotation; scope notes.	The meaning of a term, fixed by providing a context for its interpretation, both positive (denotation) and limiting/negative (scope), using either other defined terms, or natural language (primitive semantics).
Serialisation		XML: a format for providing structured information in a uni-directional transfer between computers.
Set		1. ONIX: a closed collection of products; 2. LIDO: a composite comprising related elements normally describing one entity or coherent aspect of an entity.
Sub-element		XML: an element contained within another element to form a hierarchy. The sub-element could be another container or could contain data (data element).

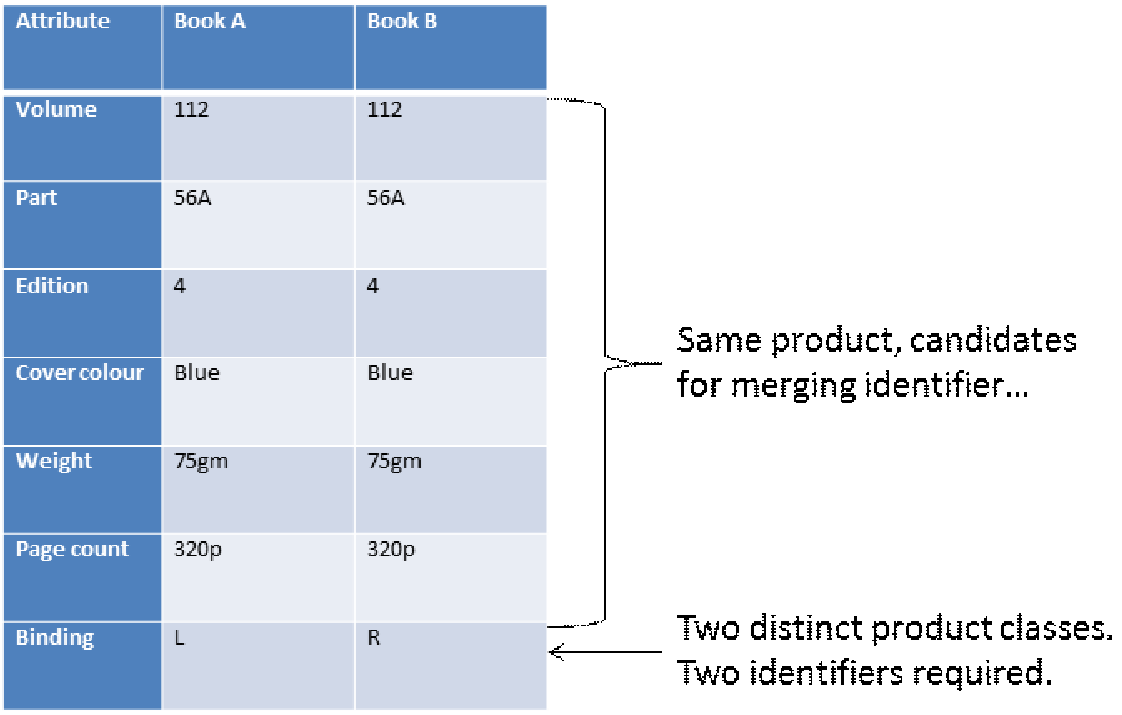
Term	Synonyms	Definition (illustrative)
Subsumption		Definition of terms by the total inclusion of their definition in other, more general terms.
Typed [data]		Data defined by a precise term. Definition can include the type of encoding used to express the data as well as its scope and denotation.
UML [Unified Modelling Language]		A comprehensive set of diagram conventions for modelling systems. This report quotes several UML-like class diagrams but only makes use of the aspects describing hierarchies and cardinalities.
Wrap		LIDO: a container for several sets of the same type of data, but for distinct entities or aspects of entities.
XML [eXtensible Markup Language]		A syntax and grammar used to define structured documents.
XPath		A language designed to reference entities of data within an XML document, as well as specifying conditions, functions and other operations on those entities.
XSD [XML Schema Description]	XML Schema	An XML language that prescribes the structure for XML documents.
XSLT [XML StyLe Sheet Transformations]	XSL	An XML language used to define transformations of one type of XML document to another.

17 APPENDIX 2 – MAPPING COMMERCIAL DATA TO CULTURAL HERITAGE SCHEMAS

This appendix details technical issues arising from the problem of integrating data that originates in two complementary but very different sectors. The issues are presented here in order of decreasing abstraction, going from the very general to the more specific, and possible solutions will be highlighted in each case.

17.1 DIFFERENT KINDS OF ENTITY DESCRIBED BY COMMERCIAL SECTOR AND HERITAGE SECTOR DATA

The main problem in mapping from commercial sector data to cultural heritage schemas is that the primary entities described are different in each scheme. This is explained in detail in D4.1. The commercial sector is centred on generic products which can be sold in many instances once created; these are abstract entities called "manifestations", consisting of the relevant characteristics of all functionally identical[142] products. As explained by Paskin (2004), "[w]e can always add another attribute to make two "like" things "unlike"… No set of metadata elements is definitive for all purposes… For a machine, "for the purpose of" = "class having this set of attributes"".



Attribute	Book A	Book B
Volume	112	112
Part	56A	56A
Edition	4	4
Cover colour	Blue	Blue
Weight	75gm	75gm
Page count	320p	320p
Binding	L	R

Same product, candidates for merging identifier…

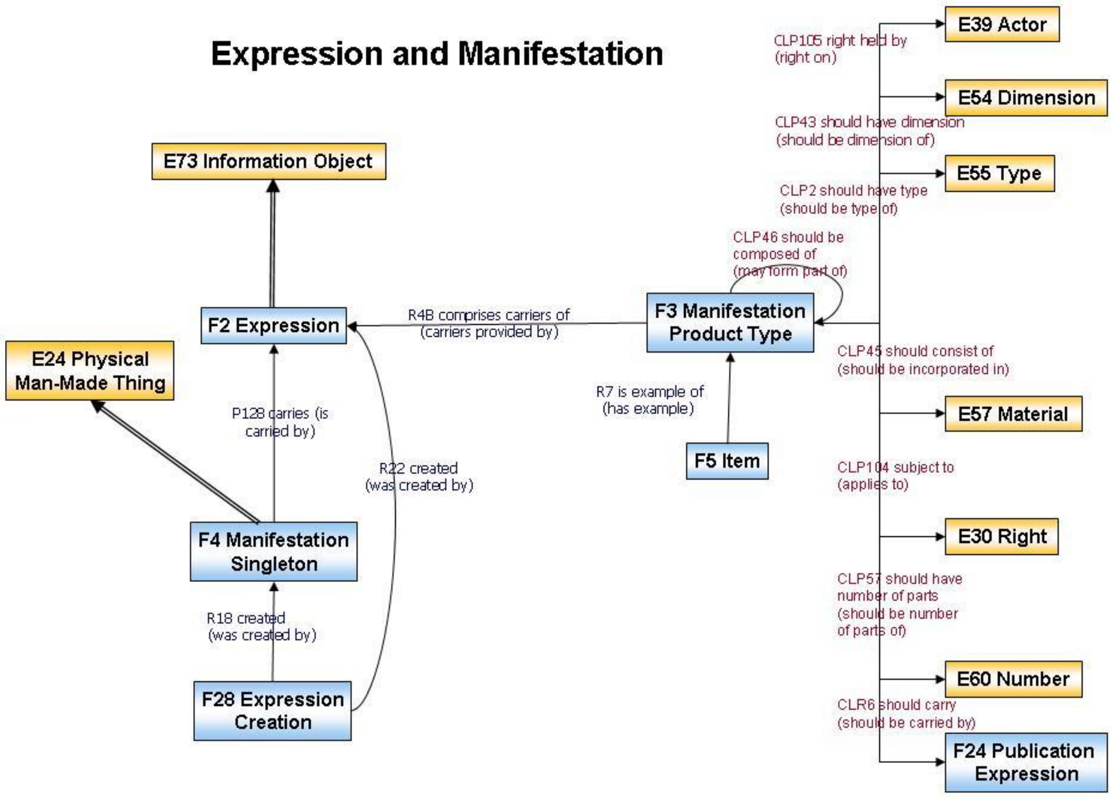
Two distinct product classes. Two identifiers required.

This type of identification process is used for the registration of product in identifier registries such as ISBN agencies, the EIDR registry and ISWC and ISRC registries. The set of attributes used to ensure that duplicate products are not registered with the same identifier, its reference descriptive metadata[143] is illustrative of the requirements for unambiguous identification within the relevant content sector. Once the product information is released beyond the "curatorial environment" of the product's originating company, these attributes make the product recognisable for the whole supply chain.

Note that as per the FRBRoo analysis, the attributes of a product, as reproduced at industrial scale, are specified based on an exemplary "proof" of the final product design (F4 Manifestation Singleton) plus the contribution of the publisher (F24 Publication Expression) and are predicated of the individual items reproduced by a standardised process only in the sense that all items identified as being "this product"

[142] That is, identical for the purposes of identification within the supply chain; see Paskin, N. (2004).
[143] See ISO TC46 SC9. (2006). *Use cases for interoperability of ISO TC46 SC9 identifiers.* Available at http://www.collectionscanada.gc.ca/iso/tc46sc9/docs/sc9n417.pdf

should have those properties inherited by the class (F3 Manifestation Product Type). This stage of the FRBRoo analysis is characteristically kept within the commercial sector publishing organisation(s) and is effectively opaque to the heritage sector. See the diagram below for the full process of defining the Product Type.

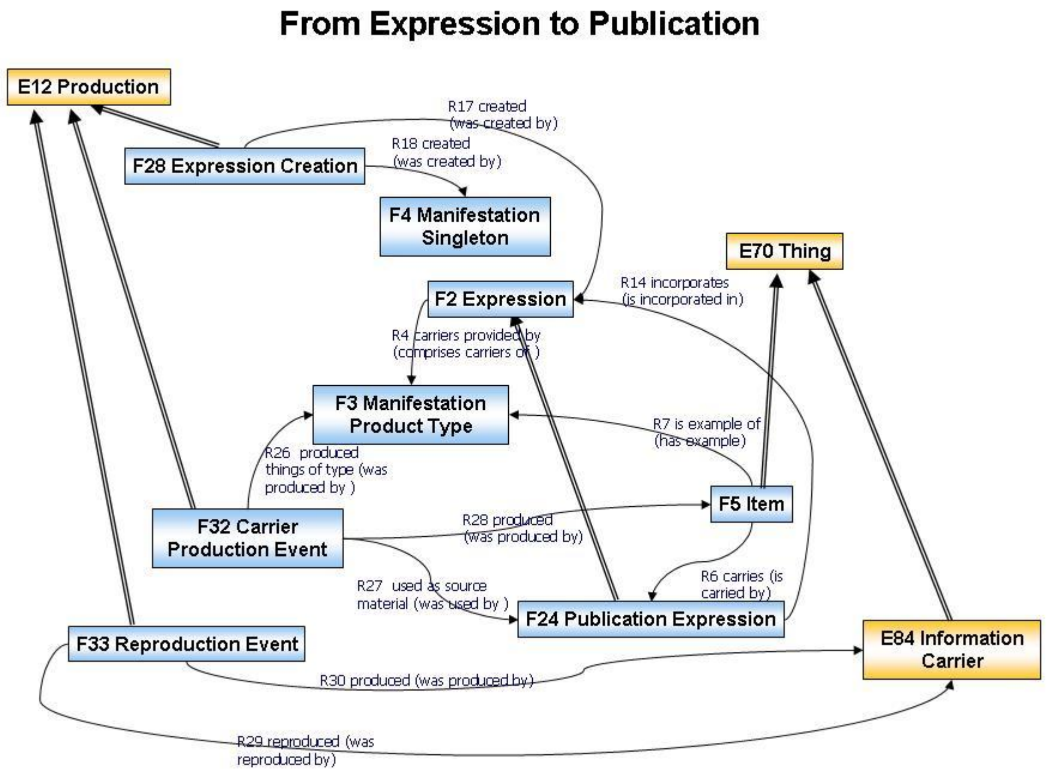

Expression and Manifestation
E73 Information Object
F2 Expression
E24 Physical Man-Made Thing
P128 carries (is carried by)
R22 created (was created by)
F4 Manifestation Singleton
R18 created (was created by)
F28 Expression Creation
R4B comprises carriers of (carriers provided by)
F3 Manifestation Product Type
R7 is example of (has example)
F5 Item
CLP105 right held by (right on)
E39 Actor
CLP43 should have dimension (should be dimension of)
E54 Dimension
CLP2 should have type (should be type of)
E55 Type
CLP46 should be composed of (may form part of)
CLP45 should consist of (should be incorporated in)
E57 Material
CLP104 subject to (applies to)
E30 Right
CLP57 should have number of parts (should be number of parts of)
E60 Number
CLR6 should carry (should be carried by)
F24 Publication Expression

FRBR object-oriented model: relating expression and publication (manifestation)

The final step of the FRBRoo model above explains the process of cataloguing the product based on one item, or instance of the product type. This is a point of contact between the heritage and commercial sectors, especially since in the networked environment, current business research and best practice indicate that “item in hand” inspection and correction of metadata is essential; the kind of resource-based cataloguing that libraries carry out has been explicitly recognised as useful for the publishing metadata supply chain[144]. In practise in the heritage world, cataloguing will happen as one discreet step, as shown in FRBRoo; in commercial contexts metadata will be built up throughout the supply chain (see D4.1, section 5.3.4).

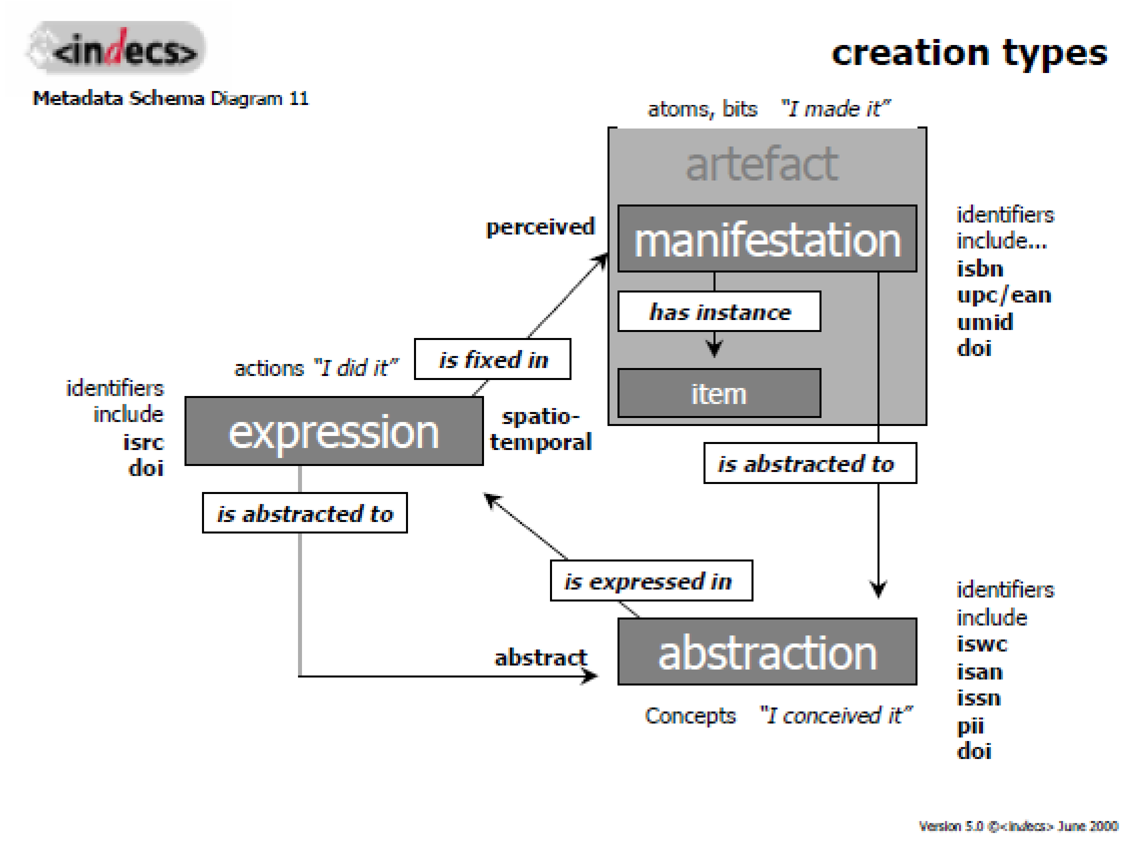
The next diagram from FRBRoo shows both how the two Manifestation entities relate to Items corresponding to actual copies of *e.g.* a published book, movie release on DVD or download, music or photo file:

[144] See Magellan Media Consulting, 2012.


From Expression to Publication
E12 Production
F28 Expression Creation
R17 created (was created by)
R18 created (was created by)
F4 Manifestation Singleton
E70 Thing
F2 Expression
R14 incorporates (is incorporated in)
R4 carriers provided by (comprises carriers of)
F3 Manifestation Product Type
R7 is example of (has example)
R26 produced things of type (was produced by)
F5 Item
F32 Carrier Production Event
R28 produced (was produced by)
R6 carries (is carried by)
R27 used as source material (was used by)
F24 Publication Expression
F33 Reproduction Event
E84 Information Carrier
R30 produced (was produced by)
R29 reproduced (was reproduced by)

FRBR object-oriented model: event description of publication based on a creative expression

Note that here, the F3 Manifestation Product Type refers to a conceptual class ("types" are simply meta-CRM lists of properties), whereas the F5 Item class is a class of individual physical copies of the book, film, recording or photo. Contrast this with the apparently ambiguous treatment of "Manifestation" in indecs, where it appears to be the aggregate set or collection of all the items:



indecs model of creative expressions, manifestations, and abstractions derived from them both

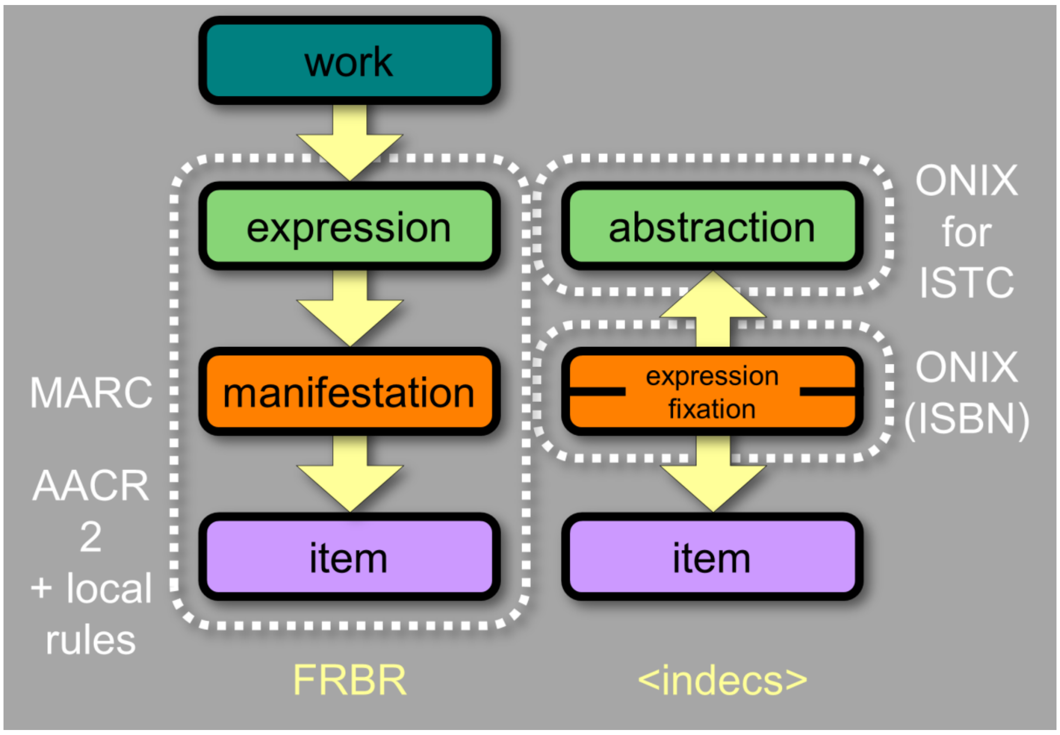
This is a convenient optimisation (flattening, or denormalisation) commonly used in metadata formats for mass-produced creative content where the item in hand typically inherits almost all properties of interest from its product type. In the case of ONIX for Books, this is clearly seen when an ONIX record is expressed as RDF by making explicit each entity and relation. In the illustrative RDF/XML below[145], it can be seen that the "product" itself is identified by the ONIX <RecordReference> which is then linked to any published identifiers the book may have (in this case, an ISBN). The missing link in this chain of data is simply the meta-CRM statement linked to a declaration that this is a product type, rather than a collection of specific (albeit functionally identical) items.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:onix="http://ns.editeur.org/onix/3.0/reference/">
  <rdf:Description rdf:nodeID="uk.co.harpercollins.onix.product.40366">
    <onix:ProductIdentifier rdf:datatype="http://ns.editeur.org/onix/codelists/5#15">
                9780007232833</onix:ProductIdentifier>


    <onix:ProductForm rdf:resource="http://ns.editeur.org/onix/codelists/150#BC" />
    <onix:Contributor>
        <rdf:Description onix:PersonNameInverted="Sjöwall, Maj">
<onix:ContributorRole rdf:resource="http://ns.editeur.org/onix/codelists/17#A01" />


        <onix:NameIdentifier
rdf:datatype="http://ns.editeur.org/onix/codelists/44#16">0000000121479135</onix:NameIdentifier>
```

[145] Example RDF/XML for illustrative purposes only. Taken from presentation by Bell, G. (2012).

```
      </rdf:Description>
    </onix:Contributor>
  </rdf:Description>
</rdf:RDF>
```

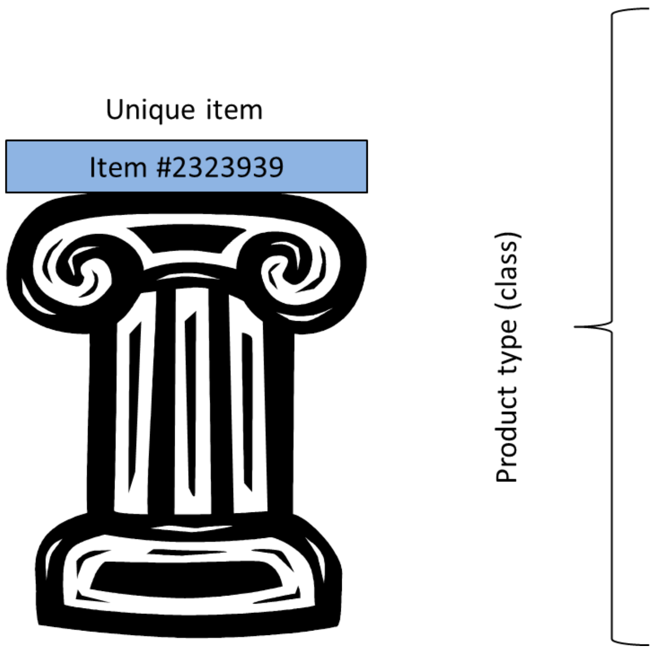
This type of optimisation is by no means unique to commercial sector metadata. In fact, it is identically used in MaRC to describe books, where they are equally interesting for their F2 Expression(s), F24 Publication Expression (together corresponding to the FRBR and ONIX Manifestation), but also F5 Item, as in the diagram below.



Comparison of FRBR (library cataloguing) and indecs (commercial product information) views of books

However, to integrate information about books on the fringes of the "uniqueness spectrum" (see D4.1, section 4.3), such as rare books and special collections, libraries may have difficulties with the MaRC and cataloguing rules paradigm, because it does not always integrate the more dynamic modelling expressed to some degree in ONIX (as shown by the separation of manifestation into expression and fixation processes) and more fully for individual items in the CIDOC-CRM.

In contrast, the cultural heritage sector primarily documents unique objects and their provenance:



Fundamental differences in heritage and commercial sector data models: item versus class description

In the above diagram, note that all of the blue boxes represent objects that are to the book trade substitutable except for their actual location, price, ownership and so forth (*e.g.* a particular book). Any of the product numbered items on the right is replaceable or substitutable with any of the other identical products in the "pile". The precise historical events that led up to these functionally identical books being available (authoring, publishing, printing, distribution, retail) is almost totally irrelevant to the reader; only the "content" matters. Perhaps some slight differences in the physical condition of the item affect the price but these are not considered part of the standard product; it is primarily a means of getting the content to the reader. This represents the commercial data view.

On the left, the unique single copy of the product (placed on a pedestal) represents the cultural heritage view; *this item* is the entity being described, including any "defects" that distinguish it from the otherwise identical objects on the right, and the provenance of the item, its history, is a central aspect:

"A cultural-historical research space… provides access to primary knowledge about objects and in archival material. This information is prior to having a subject in the library sense. A museum object is more like an illustration or witness of the past, than information in its own right." (Doerr, 2010).

So, in the case that the blue boxes represent a book, the "content" of the book is, in the cultural heritage view secondary, or at least, on the same level of interest as the unique historical journey *this copy* took in reaching our observer. In this case, the item is numbered and identified as an item, not a representative of a class, and is not replaceable by another similar item.

Of course this comparison uses extreme cases to make a point; rare or limited print-run books (for example) are often more or less unique. Museums do collect many items that are not entirely unique. The general difference in focus holds for the normal case in each sector[146].

17.1.1 Solutions – exemplary item cataloguing or class property mapping?

The simplest and most obvious solution is to map the properties that belong to the manifestation to a schema describing a hypothetical, exemplary item (which could in theory exist, selected arbitrarily from the pile of identical objects in the above diagram and "put on a pedestal" as a representative of its class!). The only condition would be that it represents *all* the necessary characteristics needed to belong to the manifestation, and no less (a fair assumption given the commercial, mass-production context and

[146] In the music sector especially, the problem could in principle become even more complex because abstract "release" (content listing or manifest) entities can be described for a range of different but strongly related products.

reasonable level of curation and preservation of the individual item). In fact, this concept already exists in FRBR(oo)[147]. Something like this approximation takes place in reverse when an ONIX record is used as the basis for a MaRC catalogue record, before the item-in-hand check is done on the exemplary copy. However, this approach is only applicable in the context of actual repository collections of items, such as libraries and film archives. It does not fully address the need to ingest product data originating directly from the commercial sector.

The approach taken in this report, after experimenting with many examples of product data and exploring the capacity of the LIDO schema and its related data models in the CIDOC-FRBR domains, was to recommend the use of a LIDO-CRM mapping based on the class properties introduced by the FRBRoo and meta-CRM working drafts. The LIDO schema already allows for such use in its inclusion of the record category field, which in principle could take values from both the physical and conceptual sides of the CIDOC-CRM and hence include F3 Manifestation Product Type. When this category is selected for the LIDO record, the LIDO elements should be interpreted as referring to class properties inherited from the F2 Expression(s) and F24 Publication Expression, with the inclusion of this extra triple in each CRM expression (but otherwise identical to the semantics of a LIDO record for the item – minus any distinguishing features, of course!).

Note that this report has experimented and made recommendations on the basis of the available tools and frameworks – principally LIDO and MINT. Significant extensions and developments to both of these have been recommended. Given further scope and resources it would be possible to adopt more general methods such as building on the work of VMF to model common properties of commercial products relevant to the specific use case(s).

17.2 RELATIVE FREQUENCY OF TRANSACTIONS IN EACH SECTOR

A second, related difference between the commercial and cultural heritage sectors is the relative frequency of transactions, or more generally, events involving the objects involved. This is due to the relative value placed on items in themselves and as witnesses to events (provenance), versus their value as carriers of information content. Thus instead of details of the finding, collection and (perhaps a few) transfers of custodianship) in the heritage sector, one finds details of the intellectual property rightsholder(s), potential markets where sales are licensed, suppliers who make the product available to end customers, and links to order or directly access content – and in each case, there are likely many, rather than few.

This difference is important for the Linked Heritage use case primarily because whereas a heritage object is normally held in custody and curated by one and the same institution, analogously a commercial product may be available for retail from a number of suppliers, including the publisher themselves. It is not immediately clear who is the “curator” of the product itself, as in practice, the object of interest for trade offers and agreements is, at least early in the supply chain, a right or license regarding the intellectual content of the product (supply of physical items is somewhat distinct from this). The one fixed (category of) rightsholder may be the creator(s) themselves – however, even this distinction can vary depending on a given jurisdiction’s balance of statutory and contractual rights.

Because of this, aggregating product data in a heritage context simultaneous offers more and less “access” to the product. More, because as well as linking to information about the product in context from its metadata, previews, extracts and the product itself may be obtained. Less, because in order to provide this link it is necessary to impose the somewhat artificial concept of a product “repository” on the data, which may impede the business case for any data contribution at all. Here the interdependence of rights and descriptive metadata becomes clear[148]. The territoriality of retail sales rights is also in play here – some retailers will not be able to offer a product to customers in certain geographic locations, and will not wish their products to be advertised through metadata to those customers, or not without a disclaimer. Neither Europeana nor the LIDO / MINT infrastructure currently support such sensitivity automatically.

[147] F4 Manifestation Singleton: R42 is representative manifestation singleton for: F2 Expression (see Bekiari, Doerr and LeBoeuf, 2009; pp125-127).
[148] For a full discussion see Rust, 1998. The key passage is found at this link: http://www.dlib.org/dlib/july98/rust/07rust.html#dependence_of_rights_metadata

In the Europeana aggregation schema, a maximum of one link is allowed to a Web page for the Digital Object (the image of the Cultural Heritage Object in context).[149] The choice of publisher or retailer link to map to this element is unclear and will depend partly on non-technical, legal and commercial factors. Hence a fuller discussion of this point will appear in D4.3. However, in the LIDO format, multiple repositories can be specified, and multiple links to a data sheet describing the object are also allowed. This is the part of LIDO used to map such retail links here, with a preference for the publisher or other releasing company, since it is closer to the "origin" of the product. However, this does not solve the problem above since in practice retail links may not be offered by publishers and the question of which retail channel to prefer remains.

17.2.1 Solutions – multiple resolution or planned future events?

Allowing multiple access points or "repositories" for each product is a partial solution. Ultimately a new approach will be required to resolve the need to fully reflect the commercial nature of the product information aggregated here. Some existing tools and services point towards possible long-term solutions:

- Multiple resolution services like the ISBN-A.
  One way to short-cut the problem of selection one "repository" for a product would be to use an intermediary resolution service like that offered by ISBN-A (as described in D4.1, section 6.3.5).
- Aggregation of sales offers via affiliates.
  In D4.1 section 8.2.4 the UK service Findanyfilm.com was described. This website aggregates descriptive metadata about films, actors, and directors, and also collates cinema release and retail recording availability information with links to book tickets or buy recordings. These links are provided by third-party affiliate services that independently aggregate offers from a range of partners, updating and managing the legal and commercial aspects. This type of solution would potentially integrate with a multiple resolution service like ISBN-A; otherwise it would be necessary to reproduce this functionality within the aggregator platform along with other types of updates.
- Extension of data integration schema to include planned events.
  In the case that retail link management were managed within the central data aggregator, the core data model would need to be extended to explicitly include planned future events. These represent the "offers" common in commercial metadata and could perhaps be modelled in two distinct ways:
  - As CIDOC-CRM E29 Plan entities implemented somehow in a new extension the LIDO schema. The E29 Plan would represent an intermediate stage between the product itself and the planned sale event, based on the existing event structure in LIDO. However, since the sale of items is based on instances of the product, not the product itself, the semantic chain behind this structure would be long and complex (though perhaps not necessarily seen in the LIDO extension itself). Also, to model *e.g.* ebook usage constraints it might be necessary to add an extra "rights acquisition" event type with added sub-events to describe categories of allowed activity (viewing, downloading, printing, lending *etc.*). The same would apply to other digital assets.
  - The sale could be modelled simply as an event within the product information itself, with the semantic mapping taken from meta-CRM to indicate this event "should" or "usually" takes place. Usage rights information for electronic products would still need to be modelled as sub-events.
- Integrate within existing ecommerce and rights data frameworks.
  As mentioned above, categories of use for ebooks and other digital products are best modelled in terms of rights to perform certain activities in relation to the product. In the most general analysis, for traditional analogue media too, sales and use (for example, format shifting or copying) can be modelled this way. One current cross-media project, the Linked Content Coalition, aims to make integration of rights information from any content sector interoperable and enable more automated transactions[150]. This framework would potentially support the data

[149]ESE V3.4 schema available at http://pro.europeana.eu/documents/900548/dc80802e-6efb-4127-a98e-c27c95396d57
[150] For full details of the LCC see http://www.linkedcontentcoalition.org/The_Project.html

modelling needed for expressing sales offers within LIDO, and also perhaps assist with aggregating affiliate links via a partner service. Some of the use cases examined by the LCC[151] overlap with the need to provide commercially acceptable technical solutions within a product data aggregation context.

17.2.2 Modelling note – publisher as "repository" for product type?

It should be noted that the most obvious default for the "repository" of a product type is the publisher, and thus probably in the absence of a dynamic solution like those outlined above, publisher information should be chosen for this part of any mappings to cultural heritage formats. The publisher or other "releasing" company is usually (though not always) the registrant of the product for identification purposes, and what is registered is actually the product type information, or some minimum subset thereof. Publishers do not always act as commercial sources for their products, appointing distributors to deal with retail or wholesale enquiries; however, as maintainers of the primary *product information* as it relates to the identification of a unique product, all the other supply chain partners are depend on them.

17.3 SEMI-STATIC ARCHIVES OR DYNAMIC DATAFLOWS

Third, partly because of the difference in transaction frequency noted, but also for many other legal and commercial reasons, the concept of a "dataset" in the two viewpoints is substantially different.

Whereas in the heritage sector events involving the object of interest are viewed with academic detachment and possibly documented by one curator several times drawing on sources with alternate viewpoints of the same event, in the commercial world, the documentation of events is much more focussed on accounting for transactions involving the "curator" of the data *at that point in the supply chain.* So, not only may one record for a given product include data contributed by various partners in a supply chain, there may also be divergent or contradictory records for the same product, even if the records are produced simultaneously. These are not of scholarly or cultural interest; instead, business rules will be applied to produce a single authoritative product record for use in transactions.

Types of update message may include:

- Change of status;
- Changes of descriptive detail;
- Changes in marketing collateral;
- Changes in copyright ownership, sales rights and price;
- Change in publishing rights;
- Change in retail rights;
- Corrections to any of the above;
- Deletion of the record.

Indicators for the update notification type may occur within the message itself, and so be treated by a straightforward aggregator as simply another piece of data, even if they are intended as triggers for processing or requesting other messages. This also means that it will only be possible to directly map a subset of possible messages, as some combinations of elements will likely not occur in straightforward notification messages.

[151] See use cases summary at http://www.linkedcontentcoalition.org/uploads/EPC_Big_Idea_The_Answer_to_the_Machine_UseCases.pdf

17.3.1 Solution – business rules and revised technical platform?

As noted above, integration of sales rights and retail availability would require significant development of the data model and aggregator software. Part of this development would certainly include the facility to receive and integrate updates to metadata records, and apply business rules (beyond validating received updates against the message schema) to distinguish between contradictory information for the same product (for example, testing against dates for valid use of book or DVD cover photos; territorial restrictions on sales rights; “windowing” releases of recordings in different formats and territories depending on dates, and with different marketing collateral and localised titles).

18 APPENDIX 3 –CIDOC-CRM AND FRBROO MODELS FOR PRODUCTS

From an email on Wed, 4 Jan 2012 from Patrick Le Boeuf to the CRM Special Interest Group mailing list. The expressions below are in the form of “sentences” which typically consist of an entity (code beginning F or E) linked to another entity via a relator or property (with a code starting P, R or CLR). More examples are found in the CIDOC-CRM and FRBRoo documentation found at http://www.cidoc-crm.org/

“Yes, it is quite possible to use a combination of FRBRoo and CIDOC CRM to model commercially available reproductions of unique objects. Possible paths include:

a) "replicas of iconic artefacts:"

E22 Man-Made Object [= the reproduced unique artefact] P16B was used for(P16.1 mode of use: E55 Type {source for reproduction}) F30 Publication Event.

F30 Publication Event R24 created F24 Publication Expression [= the set of signs present on the commercial product, including its packaging].

F24 Publication Expression CLR6B should be carried by F3 Manifestation Product Type [= the commercial product].

F24 Publication Expression P130 shows features of (P130.1 kind of similarity: E55 Type {commercialized replica}) E22 Man-Made Object [= the reproduced unique artefact].

F24 Publication Expression R27B was used by F32 Carrier Production Event [= the industrial process through which all individual exemplars of the product are made].

F32 Carrier Production Event R28 produced F5 Item [= each individual physical exemplar of the commercial product].

b) "prints of photos of paintings:"

E22 Man-Made Object [= the photographed painting] P16B was used for (P16.1 mode of use: E55 Type {photographed item}) F29 Recording Event.

F29 Recording Event P2 has type E55 Type {making photographs}.

F29 Recording Event R21 created F26 Recording [= the set of signs present on the photograph of the painting that was used as source for the publication].

R26 Recording P2 has type E55 Type {photograph}.

F26 Recording R14B is incorporated in F24 Publication Expression [= the set of signs present on the commercial product, including its packaging].

F24 Publication Expression CLR6B should be carried by F3 Manifestation Product Type [= the commercial product].

F24 Publication Expression P130 shows features of (P130.1 kind of

similarity: E55 Type {commercialized photograph}) E22 Man-Made Object [= the photographed painting].

F24 Publication Expression R27B was used by F32 Carrier Production Event [= the industrial process through which all individual exemplars of the product are made].

F32 Carrier Production Event R28 produced F5 Item [= each individual physical exemplar of the commercial product].

c) "compilations of sound recordings from archives:"

F26 Recording [= the content of sound archives] R14B is incorporated in F24 Publication Expression [= the set of signs present on the commercial product, including its packaging].

F24 Publication Expression R27B was used by F32 Carrier Production Event [= the industrial process through which all individual exemplars of the product are made].

F32 Carrier Production Event R28 produced F5 Item [= each individual physical exemplar of the commercial product].

d) Exhibition catalogues and educational DVDs are modelled exactly the same way as any book and any DVD, see FRBRoo.

19 APPENDIX 4 – ONIX FOR BOOKS 3.0 TO LIDO MAPPING

The table presented here lists only the XSLT for the main element and attribute mappings. There are 86 value "maps" within the full XSLT which almost all reproduce entire ONIX for Books code lists[152] (some only use parts where relevant) for mapping codes to concept labels. Examples of these "maps" are shown in the body of the report where relevant. The full XSLT file published with this report contains all of the full implementable value mappings. Eventually these internal XSLT value mappings should be replaced by SKOS integration via the TMP.

The XSLT below is presented almost exactly as output from the MINT platform. Linked Heritage partners and others who wish to examine the mapping in MINT can request permission to receive access from EDItEUR. The XSLT below has occasionally edited to remove whitespace (especially around sequences of element-closing tags) or truncated to schematically represent whole sections of the mapping structure with representative top-level tags (as in the discussion of LIDO in section 8). The listing has been distributed across the table below sometimes pragmatically by section length, and as often as possible, to follow the logical structure of the XSLT templates, but always following the LIDO schema outline structure.

19.1 READING SYNTACTIC AND CONDITIONAL MAPPINGS IN XSLT

At first glance the XLST below appears very complex, but despite its extreme length it uses only a small number of the possible XSLT elements, and most repeats the same combinations of these elements.

19.2 XSLT VARIABLES USED TO REPRESENT ONIX CODE LISTS

As noted in section 9.2, fixed XSLT variable "maps" are used to represent each single occurrence of an ONIX code list in the mapping script. This table relates the maps to code lists they originate in and notes where and why a list is used for many "map" variables. A large number of the code lists are used precisely twice because they classify not only the direct object of description but also the related product (LIDO "work/object").

"Map" number	ONIX Code list	Comments
0	5	Product identifier type code – "00" (proprietary) not included as not a published ID.
2	78	
4	150	Product form – used as lido:objectWorkType and lido:classification map.
6	150	
8	175	Product form detail – used as lido:objectWorkType and lido:classification map.

[152] See http://www.editeur.org/ONIX/book/codelists/current.html for the full set of ONIX code lists.

“Map” number	ONIX Code list	Comments
10	81	
12	91	Country code – used for all LIDO elements requiring country.
14	148	
16	121	
18	74	
20	28	
22	31	
24	30	
26	29	
28	28	
30	81	
32	152	
34	21	
36	79	
38	98	
40	98	
42	99	
44	76	
46	176	
48	196	
50	140	

“Map” number	ONIX Code list	Comments
52	148	
54	79	
56	2	Product composition – used for lido:classification and lido:recordType.
58	33	
60	79	
62	153	
64	25	
66	48	
68	48	
70	50	
72	44	Name code type – used for all instances of lido:actorID and legalBodyID.
74	18	Name / organisation type – used for all instances of lido:nameActor.
76	18	
78	18	
80	18	
82	18	
84	18	
86	18	
88	18	
90	18	
92	18	

“Map” number	ONIX Code list	Comments
94	18	
96	151	Contributor place relator – used to select content for lido:nationalityActor.
98	151	
100	49	
102	91	
104	151	
106	17	
108	33	
110	155	Content date role – partly used to specify @lido:type for related content dates.
112	155	
114	156	
116	41	
118	91	
120	44	
122	45	
124	44	
126	91	
128	44	
130	26	
132	26	
134	13	

"Map" number	ONIX Code list	Comments
136	148	
138	5	
140	98	
142	98	
144	99	
146	76	
148	176	
150	196	
152	140	
154	184	
156	5	
158	178	
160	150	
162	51	
164	5	
166	164	
168	2	
170	44	
172	44	
174	44	

19.3 ONIX 3.0.1 TO LIDO MAPPING: FULL XSLT SCRIPT

LIDO section	LIDO subsection	XSLT	Comments

Template	lidoWrap	`<xsl:template match="/">`
		`  <lido:lidoWrap>`
		`    <xsl:apply-templates select="/onix:ONIXMessage/onix:Product"/>`
		`  </lido:lidoWrap>`
		`</xsl:template>`
		`<xsl:template match="/onix:ONIXMessage/onix:Product">`
		`  <lido:lido> … ` **`[rest of XSLT here!] … </lido:lido>`**
			The lidoWrap acts like an <ONIXMessage> wrapper for many lido records, one per ONIX product record (mapped from the <Product> element) as shown by the illustrative elements in bold. Templates are applied to each <onix:Product> and the <lido:lido> record is also generated at this level.

Template	@relatedencoding	`<xsl:attribute name="lido:relatedencoding">http://ns.editeur.org/onix/3.0/reference</xsl:attribute>`	The (currently) fictitious URI for the ONIX for Books namespace is used to specify the original schema of this data. This may be updated with a release of the ONIX properties as URIs.

	lidoRecID	`<lido:lidoRecID>`
		`  <xsl:attribute name="lido:type">LINKED HERITAGE</xsl:attribute>LINKED HERITAGE:000000</lido:lidoRecID>`
			This ID is generated automatically as an identifier for the new aggregator record in LIDO. The ONIX record sender's original identifier for the input record is captured in the lido:recordWrap later in the mapping.

LIDO section	LIDO subsection	XSLT	Comments

	objectPublishedID	`<xsl:for-each select="onix:ProductIdentifier/onix:IDValue">`
		`  <lido:objectPublishedID>`
		`    <xsl:attribute name="lido:type">`
		`      <xsl:for-each select="../onix:ProductIDType">`
		`        <xsl:if test="position() = 1">`
		`          <xsl:variable name="idx1" select="index-of($map0/map, normalize-space())"/>`
		`          <xsl:choose>`
		`            <xsl:when test="$idx1 > 0">`
		`              <xsl:value-of select="$map0/map[$idx1]/@value"/>`
		`            </xsl:when>`
		`            <xsl:otherwise>`
		`              <xsl:value-of select="."/>`
		`            </xsl:otherwise></xsl:choose></xsl:if></xsl:for-each></xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectPublishedID>`
		`</xsl:for-each>`
			This part of the lido:lido section creates one <lido:objectPublishedID> for every onix:ProductIdentifier/onix:IDValue. This enables multiple “object” identifiers per lido record, corresponding exactly to the situation in ONIX.
			The lido:type attribute is generated from the <ProductIDType> code in the same composite by transforming the code using map0, a list of registered ID types (see section 6.4.3 for full description of this map).

LIDO section	LIDO subsection	XSLT	Comments

	category	`<lido:category>`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">URI</xsl:attribute>http://www.cidoc-crm.org/crm-concepts/F3</lido:conceptID>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>F3 Manifestation Product Type</lido:term>`
		`</lido:category>`
			This element is used to specify the type of CIDOC-CRM entity described by the lido record.
			This report recommends that this value for the lido:conceptID here should be used to trigger the (proposed) F3 Manifestation Product Type CRM mapping for the LIDO elements, transforming the LIDO record into a description of an identifiable conceptual class with inherited type properties, rather than a unique physical object (see Appendix 2, section 18 for a full discussion).

LIDO section	LIDO subsection	XSLT	Comments

	[default languages]	`<lido:descriptiveMetadata>`
		`<xsl:attribute name="xml:lang">en</xsl:attribute>`
		`… [rest of lido section] … <lido:descriptiveMetadata>`
		`<lido:administrativeMetadata>`
		`<xsl:attribute name="xml:lang">en</xsl:attribute>`
		`… [rest of lido section] … </lido:administrativeMetadata>`
			These attributes actually appear at the relevant places in the overall XSLT template. They are included here because MINT allows them to be set at the “template” level.
			Here the default language is set to English, but in ONIX this is left unspecified. This report recommends MINT be developed to allow this attribute to be set for each input file upload.

Classification	Work Type	`<lido:objectWorkTypeWrap>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductFormDetail">`
		`              <lido:objectWorkType>`
		`                **<xsl:attribute name="lido:sortorder">1</xsl:attribute>**`
		`                <xsl:for-each select=".">`
		`                  <lido:conceptID>`
		`                    <xsl:attribute name="lido:type">local</xsl:attribute>`
		`                    <xsl:attribute name="lido:label">Product form detail code</xsl:attribute>`
		`                    <xsl:value-of select="."/>`
		`                  </lido:conceptID>`
		`                </xsl:for-each>`
		`                <xsl:for-each select=".">`
		`                  <xsl:variable name="idx3" select="index-of($map2/map, normalize-space())"/>`
		`                  <xsl:choose>`
		`                    <xsl:when test="$idx3 > 0">`
		`                      <lido:term>`
		`                        <xsl:value-of select="$map2/map[$idx3]/@value"/>`
		`                      </lido:term>`
		`                    </xsl:when>`
			<ProductFormDetail> tends to contain far more specific details than <ProductForm> below, however, it is not mandatory as <ProductForm> is. Therefore it was mapped as an additional LIDO WorkType along with <ProductForm>.
			Using the @lido:sortorder attribute it was possible to specify that it is preferred whenever it appears.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`<xsl:attribute name="lido:label">Product form detail</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term>`
		`</xsl:otherwise>`
		`</xsl:choose>`
		`</xsl:for-each>`
		`</lido:objectWorkType>`
		`</xsl:for-each>`	

```
<xsl:otherwise>
                      <lido:term>
                        <xsl:attribute
name="lido:addedSearchTerm">yes</xsl:attribute>
                        <xsl:attribute name="lido:label">Product form
detail</xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:term>
                    </xsl:otherwise>
                  </xsl:choose>
                </xsl:for-each>
              </lido:objectWorkType>
            </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductForm">`
		`              <lido:objectWorkType>`
		`                **<xsl:attribute name="lido:sortorder">2</xsl:attribute>**`
		`                <xsl:for-each select=".">`
		`                  <lido:conceptID>`
		`                    <xsl:attribute name="lido:type">local</xsl:attribute>`
		`                    <xsl:attribute name="lido:label">Product form code</xsl:attribute>`
		`                    <xsl:value-of select="."/>`
		`                  </lido:conceptID>`
		`                </xsl:for-each>`
		`                <xsl:for-each select=".">`
		`                  <xsl:variable name="idx5" select="index-of($map4/map, normalize-space())"/>`
		`                  <xsl:choose>`
		`                    <xsl:when test="$idx5 > 0">`
		`                      <lido:term>`
		`                        <xsl:value-of select="$map4/map[$idx5]/@value"/>`
		`                      </lido:term>`
			ONIX ProductForm is used for the more familiarly-named product categories, and is mandatory for all ONIX records.
			Although this source term is mandatory, in the LIDO output it is less important that the more specific ONIX ProductFormDetail so this term only maps with a @lido:sortorder of 2.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:otherwise>`
		`  <lido:term>`
		`    <xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`    <xsl:attribute name="lido:label">Product form</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:term>`
		`</xsl:otherwise>`
		`</xsl:choose>`
		`</xsl:when>`
		`</xsl:for-each>`
		`</lido:objectWorkType>`
		`</xsl:for-each>`	

		`</lido:objectWorkTypeWrap>`	
Classification	Classification	`<lido:objectClassificationWrap>`	

LIDO section	LIDO subsection	XSLT	Comments

	Classification	`<lido:classification>`
		`<xsl:attribute name="lido:type">europeana:type</xsl:attribute>`
		`<xsl:if test="(onix:DescriptiveDetail/onix:PrimaryContentType = '07') or (onix:DescriptiveDetail/onix:PrimaryContentType = '18') or (onix:DescriptiveDetail/onix:PrimaryContentType = '19') or (onix:DescriptiveDetail/onix:PrimaryContentType = '20') or (onix:DescriptiveDetail/onix:PrimaryContentType = '12')">`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>IMAGE</lido:term>`
		`</xsl:if>`
			The first of four conditional mappings that select one of the Europeana media types. Here are the conditions for “IMAGE”.
			These four all select on ONIX PrimaryContentType which is specified for use in describing ebooks.
			See the final conditional mapping below for “TEXT” to see the default type, which is also assumed for print books.

	Classification	`<xsl:if test="(onix:DescriptiveDetail/onix:PrimaryContentType = '01') or (onix:DescriptiveDetail/onix:PrimaryContentType = '02') or (onix:DescriptiveDetail/onix:PrimaryContentType = '13') or (onix:DescriptiveDetail/onix:PrimaryContentType = '03') or (onix:DescriptiveDetail/onix:PrimaryContentType = '04') or (onix:DescriptiveDetail/onix:PrimaryContentType = '21') or (onix:DescriptiveDetail/onix:PrimaryContentType = '22') or (onix:DescriptiveDetail/onix:PrimaryContentType = '23')">`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>SOUND</lido:term>`
		`</xsl:if>`
			The code list values used for the four Europeana media type mappings follow the order of presentation of the entries in ONIX code list 81, which is grouped according to these four same general types, even though the numerical codes themselves have a slightly different order. This is convenient, but also adds authority to this selection.

LIDO section	LIDO subsection	XSLT	Comments

	Classification	`<xsl:if test="(onix:DescriptiveDetail/onix:PrimaryContentType = '06') or (onix:DescriptiveDetail/onix:PrimaryContentType = '26') or (onix:DescriptiveDetail/onix:PrimaryContentType = '27') or (onix:DescriptiveDetail/onix:PrimaryContentType = '24') or (onix:DescriptiveDetail/onix:PrimaryContentType = '25') or (onix:DescriptiveDetail/onix:PrimaryContentType = '28') or (onix:DescriptiveDetail/onix:PrimaryContentType = '29') or (onix:DescriptiveDetail/onix:PrimaryContentType = '30') or (onix:DescriptiveDetail/onix:PrimaryContentType = '31')">`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>VIDEO</lido:term>`
		`</xsl:if>`
			Note that the map of code list 81 does not include codes for interactive content, such as games, or advertising, found grouped together at the end of the list.

	Classification	`<xsl:if test="(not(onix:DescriptiveDetail/onix:PrimaryContentType)) or (onix:DescriptiveDetail/onix:PrimaryContentType = '10') or (onix:DescriptiveDetail/onix:PrimaryContentType = '15') or (onix:DescriptiveDetail/onix:PrimaryContentType = '14') or (onix:DescriptiveDetail/onix:PrimaryContentType = '16') or (onix:DescriptiveDetail/onix:PrimaryContentType = '17') or (onix:DescriptiveDetail/onix:PrimaryContentType = '11')">`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>TEXT</lido:term>`
		`</xsl:if>`
		`</lido:classification>`
			This sets the default "TEXT" value if either there is no <PrimaryContentType> (the most likely case if the record is not for an ebook) or if the content type is one of the textual types.
			As one main focus of ONIX 3.0 was the move towards ebooks, it seemed essential to include this mapping.

LIDO section	LIDO subsection	XSLT	Comments

	Classificati on	`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductForm">`
		`<lido:classification>`
		`<xsl:for-each select=".">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">local</xsl:attribute>`
		`<xsl:attribute name="lido:label">Product form code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID>`
		`</xsl:for-each>`
			Builds the mapping from ONIX ProductForm to a single LIDO classification's <term>. The @type is set to "local" for the <conceptID> for this and all other LIDO classifications taken from an ONIX code list since the code lists are not used outside the ONIX format. The only exceptions are where the ONIX code list borrows from or replicates the whole a published standard (*e.g.* an ISO code list).

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select=".">`
		`<xsl:variable name="idx7" select="index-of($map6/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx7 > 0">`
		`<lido:term>`
		`<xsl:value-of select="$map6/map[$idx7]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`<xsl:attribute name="lido:label">Product form</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term>`
		`</xsl:otherwise>`
		`</xsl:choose>`
		`</xsl:for-each>`
		`</lido:classification>`
		`</xsl:for-each>`
			Builds the LIDO term for the ONIX ProductForm classification. As with all other direct uses of the ONIX code lists as classification schemes, this uses a simple map of the list's code to its concept label.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductFormDetail">`
		`  <lido:classification>`
		`    <xsl:for-each select=".">`
		`      <lido:conceptID>`
		`        <xsl:attribute name="lido:type">Local</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Product form detail code</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:conceptID>`
		`    </xsl:for-each>`
			Builds a LIDO classification conceptID from ONIX ProductFormDetail, extra to that included as an objectWorkType.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select=".">`
		`<xsl:variable name="idx9" select="index-of($map8/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx9 > 0">`
		`<lido:term>`
		`<xsl:value-of select="$map8/map[$idx9]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`<xsl:attribute name="lido:label">Product form detail</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term>`
		`</xsl:otherwise>`
		`</xsl:choose>`
		`</xsl:for-each>`
		`</lido:classification>`
		`</xsl:for-each>`
			Performs the code list code to label mapping for the <ProductFormDetail> term.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:PrimaryContentType">`
		`  <lido:classification>`
		`    <xsl:for-each select=".">`
		`      <lido:conceptID>`
		`        <xsl:attribute name="lido:type">Local</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Primary content type code</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:conceptID>`
		`    </xsl:for-each>`
			The PrimaryContent is used again as a classification even though it was earlier used to select a Europeana media type.
			The Europeana categories are much more general than the full ONIX code list so it was important to map it in full as a classification here to preserve the richer semantics.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select=".">`
		`  <xsl:variable name="idx11" select="index-of($map10/map, normalize-space())"/>`
		`  <xsl:choose>`
		`    <xsl:when test="$idx11 > 0">`
		`      <lido:term>`
		`        <xsl:value-of select="$map10/map[$idx11]/@value"/>`
		`      </lido:term>`
		`    </xsl:when>`
		`    <xsl:otherwise>`
		`      <lido:term>`
		`        <xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Primary content type</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:term>`
		`    </xsl:otherwise>`
		`  </xsl:choose>`
		`</xsl:for-each>`
		`</lido:classification>`
		`</xsl:for-each>`
			The term mapping direct from the ONIX code list as for ProductForm and ProductFormDetail.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:CountryOfManufacture">`
		`  <lido:classification>`
		`    <xsl:for-each select=".">`
		`      <lido:conceptID>`
		`        <xsl:attribute name="lido:type">ISO 3166-1</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Country of manufacture code</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:conceptID>`
		`    </xsl:for-each>`
			One of a small number of classifications with an ISO standard for its @type attribute. The ONIX code list it maps from incorporates the ISO codes and labels directly so the @type of both the LIDO conceptID and term are inherited from the ISO standard name.

LIDO section	LIDO subsection	XSLT	Comments

		`    <xsl:for-each select=".">`
		`      <xsl:variable name="idx13" select="index-of($map12/map, normalize-space())"/>`
		`      <xsl:choose>`
		`        <xsl:when test="$idx13 > 0">`
		`          <lido:term>`
		`            <xsl:value-of select="$map12/map[$idx13]/@value"/>`
		`          </lido:term>`
		`        </xsl:when>`
		`        <xsl:otherwise>`
		`          <lido:term>`
		`            <xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`            <xsl:attribute name="lido:label">Country of manufacture</xsl:attribute>`
		`            <xsl:value-of select="."/>`
		`          </lido:term>`
		`        </xsl:otherwise>`
		`      </xsl:choose>`
		`    </xsl:for-each>`
		`  </lido:classification>`
		`</xsl:for-each>`
			Maps the term from the code list label as selected by the code present in the ONIX element. Note that CountryOfManufacture is a rarely-used ONIX element but forms a useful bridge between the properties of the items (individual printed books) normally of interest in LIDO records and majority of the product type properties described by ONIX records.

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:DescriptiveDetail/onix:Collection">` `<lido:classification>` `<xsl:for-each select="onix:CollectionType">` `<lido:conceptID>` `<xsl:attribute name="lido:type">Local</xsl:attribute>` `<xsl:attribute name="lido:label">Collection type code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:conceptID>` `</xsl:for-each>`	<conceptID> mapping of ONIX CollectionType – note that the CollectionType is contained within the ONIX <Collection> composite, so this XSLT creates one LIDO classification per ONIX <Collection> and thus allowing multiple collection types to appear in the LIDO record.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:CollectionType">`
		`  <xsl:variable name="idx15" select="index-of($map14/map, normalize-space())"/>`
		`  <xsl:choose>`
		`    <xsl:when test="$idx15 > 0">`
		`      <lido:term>`
		`        <xsl:value-of select="$map14/map[$idx15]/@value"/>`
		`      </lido:term>`
		`    </xsl:when>`
		`    <xsl:otherwise>`
		`      <lido:term>`
		`        <xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Collection type</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:term>`
		`    </xsl:otherwise>`
		`  </xsl:choose>`
		`</xsl:for-each>`
		`</lido:classification>`
		`</xsl:for-each>`
			Another direct mapping of ONIX code list values.

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:DescriptiveDetail/onix:Language">` **`<xsl:if test="(onix:LanguageRole = '01') or (onix:LanguageRole = '03') or (onix:LanguageRole = '06') or (onix:LanguageRole = '07') or (onix:LanguageRole = '08') or (onix:LanguageRole = '09')">`** `<lido:classification>`	Builds the container element <classification> for several pairs of <conceptID> and <term> elements based on the ONIX <Language> composite. These are conditional on the <Language> composite referring to a language used in the product itself (in the primary textual content or elsewhere). It would be more useful to be able to specify this language role in the LIDO record but this appears to be impossible in the LIDO <classification> structure.

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:CountryCode">` `<lido:conceptID>` `<xsl:attribute name="lido:type">ISO 3166-1</xsl:attribute>` `<xsl:attribute name="lido:label">Country code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:conceptID>` `</xsl:for-each>` `<xsl:for-each select="onix:ScriptCode">` `<lido:conceptID>` `<xsl:attribute name="lido:type">ISO 15924</xsl:attribute>` `<xsl:attribute name="lido:label">Script code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:conceptID>` `</xsl:for-each>` `<xsl:for-each select="onix:LanguageCode">` `<lido:conceptID>` `<xsl:attribute name="lido:type">ISO 639-2/B</xsl:attribute>` `<xsl:attribute name="lido:label">Language code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:conceptID>` `</xsl:for-each>`	Maps the LIDO term elements for ONIX CountryCode, ScriptCode, and LanguageCode. Note the ISO standards incorporated in the ONIX code lists are used as @type values.
		`<xsl:for-each select="onix:CountryCode">` `<lido:term>` `<xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>` `<xsl:attribute name="lido:label">Country</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:term>` `</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:ScriptCode">`
		`  <xsl:variable name="idx17" select="index-of($map16/map, normalize-space())"/>`
		`  <xsl:choose>`
		`    <xsl:when test="$idx17 > 0">`
		`      <lido:term>`
		`        <xsl:value-of select="$map16/map[$idx17]/@value"/>`
		`      </lido:term>`
		`    </xsl:when>`
		`    <xsl:otherwise>`
		`      <lido:term>`
		`        <xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Script</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:term>`
		`    </xsl:otherwise>`
		`  </xsl:choose>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		(see code below)	

```
                  <xsl:for-each select="onix:LanguageCode">
                    <xsl:variable name="idx19" select="index-of($map18/map,
normalize-space())"/>
                    <xsl:choose>
                      <xsl:when test="$idx19 > 0">
                        <lido:term>
                          <xsl:value-of select="$map18/map[$idx19]/@value"/>
                        </lido:term>
                      </xsl:when>
                      <xsl:otherwise>
                        <lido:term>
                          <xsl:attribute
name="lido:addedSearchTerm">yes</xsl:attribute>
                          <xsl:attribute name="lido:label">Language</xsl:attribute>
                          <xsl:value-of select="."/>
                        </lido:term>
                      </xsl:otherwise>
                    </xsl:choose>
                  </xsl:for-each>
                </lido:classification>
              </xsl:if>
            </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:AudienceCode">`
		`  <lido:classification>`
		`    <xsl:for-each select=".">`
		`      <lido:conceptID>`
		`        <xsl:attribute name="lido:type">Local</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Audience code</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:conceptID>`
		`    </xsl:for-each>`
		`    <xsl:for-each select=".">`
		`      <xsl:variable name="idx21" select="index-of($map20/map, normalize-space())"/>`
		`      <xsl:choose>`
		`        <xsl:when test="$idx21 > 0">`
		`          <lido:term>`
		`            <xsl:value-of select="$map20/map[$idx21]/@value"/>`
		`          </lido:term>`
		`        </xsl:when>`
			Maps the LIDO conceptID and term for the free-standing ONIX <AudienceCode> element. The values are mapped directly from the ONIX code list, hence the @type value “local”.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>`
		`<xsl:attribute name="xml:lang">eng</xsl:attribute>`
		`<xsl:attribute name="lido:label">Audience</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term>`
		`</xsl:otherwise>`
		`</xsl:choose>`
		`</xsl:for-each>`
		`</lido:classification>`
		`</xsl:for-each>`
			The other attributes for the <term> of the <classification> mapped from ONIX AudienceCode. Note the @xml:lang attribute is set to English as this is the language of the ONIX code list labels.

		`<xsl:for-each select="onix:DescriptiveDetail/onix:AudienceRange">`
		`<lido:classification>`
			Creates the container <classification> element for a complex multi-part mapping from the ONIX <AudienceRange> composite, dependent on the order of its sub-elements.

		`<xsl:for-each select="onix:AudienceRangeQualifier">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>`
		`<xsl:attribute name="lido:label">Audience range qualifier code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID>`
		`</xsl:for-each>`
			Creates the <conceptID> for the ONIX AudienceRangeQualifier – the <term> is mapped several sections later due to the XSLT generated by MINT. Semantically, a local code from an ONIX code list.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test=”(position() = 1)”>`
		`  <xsl:for-each select=”onix:AudienceRangePrecision[(position() = 1)]”>`
		`    <xsl:variable name=”idx23” select=”index-of($map22/map, normalize-space())”/>`
		`    <xsl:choose>`
		`      <xsl:when test=”$idx23 > 0”>`
		`        <lido:term>`
		`          <xsl:value-of select=”$map22/map[$idx23]/@value”/>`
		`        </lido:term>`
		`      </xsl:when>`
		`      <xsl:otherwise>`
		`        <lido:term>`
		`          <xsl:attribute name=”lido:addedSearchTerm”>no</xsl:attribute>`
		`          <xsl:attribute name=”lido:label”>Audience range precision 1</xsl:attribute>`
		`          <xsl:value-of select=”.”/>`
		`        </lido:term>`
		`      </xsl:otherwise>`
		`    </xsl:choose>`
		`  </xsl:for-each>`
		`</xsl:if>`
			This conditional mapping, like the others below, tests the order of the XML elements in the source ONIX file and preserves the information the order carries by adding @label attributes in the resultant LIDO elements. The semantics of the entire <Audience> statement must be reconstructed using the ONIX Best practice guide, as the LIDO schema does not contain any syntactic structures for explicitly expressing them.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="(position() = 1)">`
		`  <xsl:for-each select="onix:AudienceRangeValue[(position() = 1)]">`
		`    <lido:term>`
		`      <xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`      <xsl:attribute name="lido:label">Audience range value 1</xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:term>`
		`  </xsl:for-each>`
		`</xsl:if>`
			Adds the appropriate @label for the target <term> of the first AudienceRangeValue in the ONIX file's XML.

		`<xsl:if test="(position() = 2)">`
		`  <xsl:for-each select="onix:AudienceRangeValue[(position() = 2)]">`
		`    <lido:term>`
		`      <xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`      <xsl:attribute name="lido:label">Audience range value 2</xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:term>`
		`  </xsl:for-each>`
		`</xsl:if>`
			Adds the appropriate @label for the target <term> of the second AudienceRangeValue in the ONIX file's XML.

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="(position() = 2)">`
		`  <xsl:for-each select="onix:AudienceRangePrecision[(position() = 2)]">`
		`    <lido:term>`
		`      <xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`      <xsl:attribute name="lido:label">Audience range precision 2</xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:term>`
		`  </xsl:for-each>`
		`</xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                <xsl:for-each select="onix:AudienceRangeQualifier">
                  <xsl:variable name="idx25" select="index-of($map24/map, normalize-
space())"/>
                  <xsl:choose>
                    <xsl:when test="$idx25 > 0">
                      <lido:term>
                        <xsl:value-of select="$map24/map[$idx25]/@value"/>
                      </lido:term>
                    </xsl:when>
                    <xsl:otherwise>
                      <lido:term>
                        <xsl:attribute
name="lido:addedSearchTerm">yes</xsl:attribute>
                        <xsl:attribute name="lido:label">Audience range
qualifier</xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:term>
                    </xsl:otherwise>
                  </xsl:choose>
                </xsl:for-each>
              </lido:classification>
            </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments
		(see code below)	

```
<xsl:for-each select="onix:DescriptiveDetail/onix:Audience">
              <lido:classification>
                <xsl:for-each select="onix:AudienceCodeValue">
                  <lido:conceptID>
                    <xsl:attribute name="lido:type">
                      <xsl:for-each select="../onix:AudienceCodeType">
                        <xsl:if test="position() = 1">
                          <xsl:variable name="idx27" select="index-of($map26/map,
normalize-space())"/>
                          <xsl:choose>
                            <xsl:when test="$idx27 > 0">
                              <xsl:value-of select="$map26/map[$idx27]/@value"/>
                            </xsl:when>
                            <xsl:otherwise>
                              <xsl:value-of select="."/>
                            </xsl:otherwise>
                          </xsl:choose>
                        </xsl:if>
                      </xsl:for-each>
                    </xsl:attribute>
                    <xsl:attribute name="lido:label">Audience code</xsl:attribute>
                    <xsl:value-of select="."/>
                  </lido:conceptID>
                </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments
		see code below	

```
    <xsl:if test="onix:AudienceCodeType = '01'">
                  <xsl:for-each
select="onix:AudienceCodeValue[../onix:AudienceCodeType = '01']">
                    <xsl:variable name="idx29" select="index-of($map28/map,
normalize-space())"/>
                    <xsl:choose>
                      <xsl:when test="$idx29 > 0">
                        <lido:term>
                          <xsl:value-of select="$map28/map[$idx29]/@value"/>
                        </lido:term>
                      </xsl:when>
                      <xsl:otherwise>
                        <lido:term>
                          <xsl:attribute
name="lido:addedSearchTerm">yes</xsl:attribute>
                          <xsl:attribute name="lido:label">Audience</xsl:attribute>
                          <xsl:value-of select="."/>
                        </lido:term>
                      </xsl:otherwise>
                    </xsl:choose>
                  </xsl:for-each>
                </xsl:if>
              </lido:classification>
            </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductContentType">`
		`  <lido:classification>`
		`    <xsl:for-each select=".">`
		`      <lido:conceptID>`
		`        <xsl:attribute name="lido:type">Local</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Product content type code</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:conceptID>`
		`    </xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		see code below	

```
                <xsl:for-each select=".">
                  <xsl:variable name="idx31" select="index-of($map30/map, normalize-
space())"/>
                  <xsl:choose>
                    <xsl:when test="$idx31 > 0">
                      <lido:term>
                        <xsl:value-of select="$map30/map[$idx31]/@value"/>
                      </lido:term>
                    </xsl:when>
                    <xsl:otherwise>
                      <lido:term>
                        <xsl:attribute
name="lido:addedSearchTerm">yes</xsl:attribute>
                        <xsl:attribute name="lido:label">Product content
type</xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:term>
                    </xsl:otherwise>
                  </xsl:choose>
                </xsl:for-each>
              </lido:classification>            </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:DescriptiveDetail/onix:Illustrated">` `<xsl:if test="(.)">` `<lido:classification>` `<xsl:for-each select=".">` `<xsl:variable name="idx33" select="index-of($map32/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx33 > 0">` `<lido:term>` `<xsl:value-of select="$map32/map[$idx33]/@value"/>` `</lido:term></xsl:when>` `<xsl:otherwise>` `<lido:term>` `<xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>` `<xsl:attribute name="lido:label">Illustrated product</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:term><xsl:otherwise></xsl:choose></xsl:for-each>` `</lido:classification></xsl:if></xsl:for-each>`	Here, and in the onix:ReligiousText mapping below, the presence of the ONIX element indicates that the product is considered to fall into that category; the element is left empty. In the absence of a URI identifying the element (and the classification concept it stands for) a simple text string without a concept ID is used here to deliver at least minimal human-readable semantic value.
		`<xsl:for-each select="onix:DescriptiveDetail/onix:ReligiousText">` `<xsl:if test="(.)">` `<lido:classification>` `<lido:term>` `<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>` `<xsl:attribute name="lido:label">Religious text</xsl:attribute>Religious text</lido:term>` `</lido:classification>` `</xsl:if>` `</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:EditionType">`
		`  <lido:classification>`
		`    <xsl:for-each select=".">`
		`      <lido:conceptID>`
		`        <xsl:attribute name="lido:type">Local</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Edition type code</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:conceptID>`
		`    </xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                <xsl:for-each select=".">
                  <xsl:variable name="idx35" select="index-of($map34/map, normalize-
space())"/>
                  <xsl:choose>
                    <xsl:when test="$idx35 > 0">
                      <lido:term>
                        <xsl:value-of select="$map34/map[$idx35]/@value"/>
                      </lido:term>
                    </xsl:when>
                    <xsl:otherwise>
                      <lido:term>
                        <xsl:attribute
name="lido:addedSearchTerm">yes</xsl:attribute>
                        <xsl:attribute name="lido:label">Edition
type</xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:term>
                    </xsl:otherwise>
                  </xsl:choose>
                </xsl:for-each>
              </lido:classification>            </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductFormFeature">` `<lido:classification>` `<xsl:if test="onix:ProductFormFeatureType = '01'">` `<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '01']">` `<lido:conceptID>` `<xsl:attribute name="lido:type">Local</xsl:attribute>` `<xsl:attribute name="lido:label">Colour of cover code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:conceptID>` `</xsl:for-each>` `</xsl:if>`	
		`<xsl:if test="onix:ProductFormFeatureType = '02'">` `<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '02']">` `<lido:conceptID>` `<xsl:attribute name="lido:type">Local</xsl:attribute>` `<xsl:attribute name="lido:label">Colour of page edge code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:conceptID>` `</xsl:for-each>` `</xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="onix:ProductFormFeatureType = '04'">`
		`  <xsl:for-each`
		`select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '04']">`
		`    <lido:conceptID>`
		`      <xsl:attribute name="lido:type">Local</xsl:attribute>`
		`      <xsl:attribute name="lido:label">Special cover material`
		`code</xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:conceptID>`
		`  </xsl:for-each>`
		`</xsl:if>`	

		`<xsl:if test="onix:ProductFormFeatureType = '05'">`
		`  <xsl:for-each`
		`select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '05']">`
		`    <lido:conceptID>`
		`      <xsl:attribute name="lido:type">Local</xsl:attribute>`
		`      <xsl:attribute name="lido:label">DVD region`
		`code</xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:conceptID>`
		`  </xsl:for-each>`
		`</xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="onix:ProductFormFeatureType = '06'"> <xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '06']"> <lido:conceptID> <xsl:attribute name="lido:type">Local</xsl:attribute> <xsl:attribute name="lido:label">Operating system requirements code</xsl:attribute> <xsl:value-of select="."/> </lido:conceptID> </xsl:for-each> </xsl:if>`	
		`<xsl:if test="onix:ProductFormFeatureType = '09'"> <xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '09']"> <lido:conceptID> <xsl:attribute name="lido:type">Local</xsl:attribute> <xsl:attribute name="lido:label">E-publication accessibility detail code</xsl:attribute> <xsl:value-of select="."/> </lido:conceptID> </xsl:for-each> </xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="onix:ProductFormFeatureType = '12'">`
		`<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '12']">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>`
		`<xsl:attribute name="lido:label">CPSIA choking hazard warning code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID>`
		`</xsl:for-each>`
		`</xsl:if>`	

		`<xsl:if test="onix:ProductFormFeatureType = '13'">`
		`<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '13']">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>`
		`<xsl:attribute name="lido:label">EU Toy Safety Hazard Warning code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID>`
		`</xsl:for-each>`
		`</xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="(onix:ProductFormFeatureType = '03') or (onix:ProductFormFeatureType = '07') or (onix:ProductFormFeatureType = '08') or (onix:ProductFormFeatureType = '30') or (onix:ProductFormFeatureType = '31') or (onix:ProductFormFeatureType = '32') or (onix:ProductFormFeatureType = '33') or (onix:ProductFormFeatureType = '34') or (onix:ProductFormFeatureType = '35') or (onix:ProductFormFeatureType = '36') or (onix:ProductFormFeatureType = '37') or (onix:ProductFormFeatureType = '40')">`
		`<xsl:for-each select="onix:ProductFormFeatureValue[(../onix:ProductFormFeatureType = '03') or (../onix:ProductFormFeatureType = '07') or (../onix:ProductFormFeatureType = '08') or (../onix:ProductFormFeatureType = '30') or (../onix:ProductFormFeatureType = '31') or (../onix:ProductFormFeatureType = '32') or (../onix:ProductFormFeatureType = '33') or (../onix:ProductFormFeatureType = '34') or (../onix:ProductFormFeatureType = '35') or (../onix:ProductFormFeatureType = '36') or (../onix:ProductFormFeatureType = '37') or (../onix:ProductFormFeatureType = '40')]">`
		`<lido:conceptID><xsl:attribute name="lido:type">Local</xsl:attribute>`	

LIDO section	LIDO subsection	XSLT	Comments
		(code below)	

```
                      <xsl:attribute name="lido:label">
                        <xsl:for-each select="../onix:ProductFormFeatureType">
                          <xsl:if test="position() = 1">
                            <xsl:variable name="idx37" select="index-of($map36/map,
normalize-space())"/>
                            <xsl:choose>
                              <xsl:when test="$idx37 > 0">
                                <xsl:value-of select="$map36/map[$idx37]/@value"/>
                              </xsl:when>
                              <xsl:otherwise>
                                <xsl:value-of select="."/>
                              </xsl:otherwise>
                            </xsl:choose>
                          </xsl:if>
                        </xsl:for-each>Â code</xsl:attribute>
                      <xsl:value-of select="."/>
                    </lido:conceptID>
                  </xsl:for-each>
                </xsl:if>
```

LIDO section	LIDO subsection	XSLT	Comments

```
                <xsl:if test="onix:ProductFormFeatureType = '01'">
                  <xsl:for-each
select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '01']">
                    <xsl:variable name="idx39" select="index-of($map38/map,
normalize-space())"/>
                    <xsl:choose>
                      <xsl:when test="$idx39 > 0">
                        <lido:term>
                          <xsl:value-of select="$map38/map[$idx39]/@value"/>
                        </lido:term>
                      </xsl:when>
                      <xsl:otherwise>
                        <lido:term>
                          <xsl:attribute
name="lido:addedSearchTerm">no</xsl:attribute>
                          <xsl:attribute name="lido:label">Color of
cover</xsl:attribute>
                          <xsl:value-of select="."/>
                        </lido:term>
                      </xsl:otherwise>
                    </xsl:choose><xsl:for-each><xsl:if>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="onix:ProductFormFeatureType = '02'">`
		`<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '02']">`
		`<xsl:variable name="idx41" select="index-of($map40/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx41 > 0">`
		`<lido:term>`
		`<xsl:value-of select="$map40/map[$idx41]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`<xsl:attribute name="lido:label">Color of page edge</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term><xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="onix:ProductFormFeatureType = '04'">` `<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '04']">` `<xsl:variable name="idx43" select="index-of($map42/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx43 > 0">` `<lido:term>` `<xsl:value-of select="$map42/map[$idx43]/@value"/>` `</lido:term>` `</xsl:when>` `<xsl:otherwise>` `<lido:term>` `<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>` `<xsl:attribute name="lido:label">Special cover material</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:term><xsl:otherwise><xsl:choose><xsl:for-each><xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                <xsl:if test="onix:ProductFormFeatureType = '05'">
                  <xsl:for-each
select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '05']">
                    <xsl:variable name="idx45" select="index-of($map44/map,
normalize-space())"/>
                    <xsl:choose>
                      <xsl:when test="$idx45 > 0">
                        <lido:term>
                          <xsl:value-of select="$map44/map[$idx45]/@value"/>
                        </lido:term>
                      </xsl:when>
                      <xsl:otherwise>
                        <lido:term>
                          <xsl:attribute
name="lido:addedSearchTerm">no</xsl:attribute>
                          <xsl:attribute name="lido:label">DVD
region</xsl:attribute>
                          <xsl:value-of select="."/>
                        </lido:term><xsl:otherwise><xsl:choose>
                  </xsl:for-each>
                </xsl:if>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="onix:ProductFormFeatureType = '06'">`
		`<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '06']">`
		`<xsl:variable name="idx47" select="index-of($map46/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx47 > 0">`
		`<lido:term>`
		`<xsl:value-of select="$map46/map[$idx47]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`<xsl:attribute name="lido:label">Operating system requirements</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term><xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="onix:ProductFormFeatureType = '09'">`
		`<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '09']">`
		`<xsl:variable name="idx49" select="index-of($map48/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx49 > 0">`
		`<lido:term>`
		`<xsl:value-of select="$map48/map[$idx49]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`<xsl:attribute name="lido:label">E-publication accessibility detail</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term></xsl:otherwise></xsl:choose></xsl:for-each>`
		`</xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		```
		        <xsl:if test="onix:ProductFormFeatureType = '12'">
		          <xsl:for-each
		select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '12']">
		            <xsl:variable name="idx51" select="index-of($map50/map,
		normalize-space())"/>
		            <xsl:choose>
		              <xsl:when test="$idx51 > 0">
		                <lido:term>
		                  <xsl:value-of select="$map50/map[$idx51]/@value"/>
		                </lido:term>
		              </xsl:when>
		              <xsl:otherwise>
		                <lido:term>
		                  <xsl:attribute
		name="lido:addedSearchTerm">no</xsl:attribute>
		                  <xsl:attribute name="lido:label">CPSIA choking hazard
		warning</xsl:attribute>
		                  <xsl:value-of select="."/>
		                </lido:term></xsl:otherwise></xsl:choose></xsl:for-each>
		        </xsl:if>
		```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="onix:ProductFormFeatureType = '13'">`
		`<xsl:for-each select="onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '13']">`
		`<xsl:variable name="idx53" select="index-of($map52/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx53 > 0">`
		`<lido:term>`
		`<xsl:value-of select="$map52/map[$idx53]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`<xsl:attribute name="lido:label">EU Toy Safety Hazard Warning</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term><xsl:otherwise></xsl:choose></xsl:for-each>`
		`</xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="(onix:ProductFormFeatureType = '03') or (onix:ProductFormFeatureType = '07') or (onix:ProductFormFeatureType = '08') or (onix:ProductFormFeatureType = '30') or (onix:ProductFormFeatureType = '31') or (onix:ProductFormFeatureType = '32') or (onix:ProductFormFeatureType = '33') or (onix:ProductFormFeatureType = '34') or (onix:ProductFormFeatureType = '35') or (onix:ProductFormFeatureType = '36') or (onix:ProductFormFeatureType = '37') or (onix:ProductFormFeatureType = '40')">`
		 `<xsl:for-each select="onix:ProductFormFeatureValue[(../onix:ProductFormFeatureType = '03') or (../onix:ProductFormFeatureType = '07') or (../onix:ProductFormFeatureType = '08') or (../onix:ProductFormFeatureType = '30') or (../onix:ProductFormFeatureType = '31') or (../onix:ProductFormFeatureType = '32') or (../onix:ProductFormFeatureType = '33') or (../onix:ProductFormFeatureType = '34') or (../onix:ProductFormFeatureType = '35') or (../onix:ProductFormFeatureType = '36') or (../onix:ProductFormFeatureType = '37') or (../onix:ProductFormFeatureType = '40')]">`
		 `<lido:term>`	

LIDO section	LIDO subsection	XSLT	Comments
		see code below	

```
                      <xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>
                      <xsl:attribute name="lido:label">
                        <xsl:for-each select="../onix:ProductFormFeatureType">
                          <xsl:if test="position() = 1">
                            <xsl:variable name="idx55" select="index-of($map54/map,
normalize-space())"/>
                            <xsl:choose>
                              <xsl:when test="$idx55 > 0">
                                <xsl:value-of select="$map54/map[$idx55]/@value"/>
                              </xsl:when>
                              <xsl:otherwise>
                                <xsl:value-of select="."/>
                              </xsl:otherwise></xsl:choose></xsl:if></xsl:for-each>
                      </xsl:attribute>
                      <xsl:value-of select="."/>
                    </lido:term></xsl:for-each></xsl:if>
              </lido:classification>
            </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductComposition">`
		`<lido:classification>`
		`<xsl:for-each select=".">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>`
		`<xsl:attribute name="lido:label">Product composition code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID>`
		`</xsl:for-each>`
		`<xsl:for-each select=".">`
		`<xsl:variable name="idx57" select="index-of($map56/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx57 > 0">`
		`<lido:term>`
		`<xsl:value-of select="$map56/map[$idx57]/@value"/>`
		`</lido:term>`
		`</xsl:when>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:otherwise>` `<lido:term>` `<xsl:attribute name="lido:addedSearchTerm">yes</xsl:attribute>` `<xsl:attribute name="lido:label">Product composition</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:term>` `</xsl:otherwise>` `</xsl:choose>` `</xsl:for-each>` `</lido:classification>` `</xsl:for-each>` `</lido:classificationWrap>` `</lido:objectClassificationWrap>`	
Identification		`<lido:objectIdentificationWrap>`	
	titleWrap	`<lido:titleWrap>`	

LIDO section	LIDO subsection	XSLT	Comments

```
            <xsl:for-each select="onix:DescriptiveDetail/onix:TitleDetail">
              <xsl:if test="(not(onix:TitleStatement)) and
(onix:TitleElement/onix:TitleWithoutPrefix) and
((not(onix:TitleElement/onix:PartNumber)) and
(not(onix:TitleElement/onix:YearOfAnnual))) and ((onix:TitleType = '00') or
(onix:TitleType = '01') or (onix:TitleType = '05')) and
(onix:TitleElement/onix:TitleElementLevel = '01')">
                <lido:titleSet>
                  <xsl:attribute name="lido:sortorder">1</xsl:attribute>
                  <xsl:if test="onix:TitleElement/onix:TitleElementLevel = '01'">
                    <lido:appellationValue>
                      <xsl:for-each
select="onix:TitleElement/onix:TitlePrefix[../onix:TitleElementLevel = '01']">
                        <xsl:value-of select="."/>
                      </xsl:for-each>Â <xsl:for-each
select="onix:TitleElement/onix:TitleWithoutPrefix[../onix:TitleElementLevel =
'01']">
                        <xsl:value-of select="."/>
                      </xsl:for-each>
                    </lido:appellationValue>
                  </xsl:if><lido:titleSet></xsl:if><xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:TitleDetail">`
		`<xsl:if test="(not(onix:TitleStatement)) and ((not(onix:TitleElement/onix:PartNumber)) and (not(onix:TitleElement/onix:YearOfAnnual))) and ((not(onix:TitleElement/onix:TitlePrefix)) and (not(onix:TitleElement/onix:TitleWithoutPrefix))) and (onix:TitleElement/onix:TitleText) and ((onix:TitleType = '00') or (onix:TitleType = '01') or (onix:TitleType = '05')) and (onix:TitleElement/onix:TitleElementLevel = '01')">`
		`<lido:titleSet>`
		`<xsl:attribute name="lido:sortorder">1</xsl:attribute>`
		`<xsl:if test="(onix:TitleElement/onix:TitleElementLevel = '01')">`
		`<xsl:for-each select="onix:TitleElement/onix:TitleText[(../onix:TitleElementLevel = '01')]">`
		`<lido:appellationValue>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each></xsl:if></lido:titleSet></xsl:if>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:DescriptiveDetail/onix:TitleDetail"> <xsl:if test="(onix:TitleElement/onix:Subtitle) and ((onix:TitleType = '00') or (onix:TitleType = '01') or (onix:TitleType = '05')) and (onix:TitleElement/onix:TitleElementLevel = '01')"> <lido:titleSet> <xsl:attribute name="lido:type">Subtitle</xsl:attribute> <xsl:attribute name="lido:sortorder">2</xsl:attribute> <xsl:if test="onix:TitleElement/onix:TitleElementLevel = '01'"> <xsl:for-each select="onix:TitleElement/onix:Subtitle[../onix:TitleElementLevel = '01']"> <lido:appellationValue> <xsl:value-of select="."/> </lido:appellationValue> </xsl:for-each> </xsl:if></lido:titleSet></xsl:if></xsl:for-each>`	
		`<xsl:for-each select="onix:DescriptiveDetail/onix:TitleDetail"> <xsl:if test="(onix:TitleStatement) and ((onix:TitleType = '00') or (onix:TitleType = '01') or (onix:TitleType = '05'))"> <lido:titleSet> <xsl:attribute name="lido:type">Display title</xsl:attribute> <xsl:attribute name="lido:sortorder">1</xsl:attribute> <xsl:for-each select="onix:TitleStatement"> <lido:appellationValue> <xsl:value-of select="."/> </lido:appellationValue> </xsl:for-each> </lido:titleSet> </xsl:if> </xsl:for-each> </lido:titleWrap>`	

LIDO section	LIDO subsection	XSLT	Comments

```
          <lido:inscriptionsWrap>
            <xsl:for-each select="onix:CollateralDetail/onix:TextContent">
              <xsl:if test="((onix:TextType = '04') or (onix:TextType = '05') or
(onix:TextType = '14')) and ((onix:ContentAudience = '00') or (onix:ContentAudience
= '03') or (onix:ContentAudience = '06')) and ((onix:Text/@textformat = '03') or
(onix:Text/@textformat = '06') or (onix:Text/@textformat = '07'))">
                <lido:inscriptions>
                  <xsl:attribute name="lido:type">
                    <xsl:for-each select="onix:TextType">
                      <xsl:if test="position() = 1">
                        <xsl:variable name="idx59" select="index-of($map58/map,
normalize-space())"/>
                        <xsl:choose>
                          <xsl:when test="$idx59 > 0">
                            <xsl:value-of select="$map58/map[$idx59]/@value"/>
                          </xsl:when>
                          <xsl:otherwise>
                            <xsl:value-of select="."/>
                          </xsl:otherwise>
                        </xsl:choose></xsl:if></xsl:for-each></xsl:attribute>
```

```
                  <xsl:for-each select="onix:Text">
                    <lido:inscriptionTranscription>
                      <xsl:value-of select="."/>
                    </lido:inscriptionTranscription>
                  </xsl:for-each>
                </lido:inscriptions>
              </xsl:if>
            </xsl:for-each>
          </lido:inscriptionsWrap>
```

LIDO section	LIDO subsection	XSLT	Comments

Identification	description	`<lido:objectDescriptionWrap>`
		`<xsl:for-each select="onix:DescriptiveDetail/onix:IllustrationsNote">`
		`<xsl:if test="(not(../onix:NumberOfIllustrations))">`
		`<lido:objectDescriptionSet>`
		`<xsl:if test="(not(../onix:NumberOfIllustrations)) and (not(../onix:AncillaryContent/onix:Number))">`
		`<xsl:for-each select=".[(not(../onix:NumberOfIllustrations)) and (not(../onix:AncillaryContent/onix:Number))]">`
		`<lido:descriptiveNoteValue>`
		`<xsl:attribute name="lido:label">Illustrations note</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:descriptiveNoteValue>`
		`</xsl:for-each>`
		`</xsl:if>`
		`</lido:objectDescriptionSet>`
		`</xsl:if>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductFormDescription">`
		`<lido:objectDescriptionSet>`
		`<xsl:for-each select=".">`
		`<lido:descriptiveNoteValue>`
		`<xsl:attribute name="lido:label">Product form description</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:descriptiveNoteValue>`
		`</xsl:for-each>`
		`</lido:objectDescriptionSet>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductFormFeature">`
		`<xsl:if test="(onix:ProductFormFeatureType = '03') or (onix:ProductFormFeatureType = '06') or (onix:ProductFormFeatureType = '07') or (onix:ProductFormFeatureType = '08') or (onix:ProductFormFeatureType = '17') or (onix:ProductFormFeatureType = '37') or (onix:ProductFormFeatureType = '40')">`
		`<lido:objectDescriptionSet>`	

		`<xsl:for-each select="onix:ProductFormFeatureDescription">`
		`<lido:descriptiveNoteValue>`
		`<xsl:attribute name="lido:label">`
		`<xsl:for-each select="../onix:ProductFormFeatureType">`
		`<xsl:if test="position() = 1">`
		`<xsl:variable name="idx61" select="index-of($map60/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx61 > 0">`
		`<xsl:value-of select="$map60/map[$idx61]/@value"/>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<xsl:value-of select="."/>`
		`</xsl:otherwise></xsl:choose></xsl:if></xsl:for-each>`
		`</xsl:attribute><xsl:value-of select="."/>`
		`</lido:descriptiveNoteValue>`
		`</xsl:for-each>`
		`</lido:objectDescriptionSet>`
		`</xsl:if></xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:DescriptiveDetail/onix:EditionStatement"> <lido:objectDescriptionSet> <xsl:for-each select="."> <lido:descriptiveNoteValue> <xsl:attribute name="lido:label">Edition statement</xsl:attribute> <xsl:value-of select="."/> </lido:descriptiveNoteValue> </xsl:for-each> </lido:objectDescriptionSet> </xsl:for-each>`	
		`<xsl:for-each select="onix:DescriptiveDetail/onix:AncillaryContent"> <xsl:if test="(not(onix:Number))"> <lido:objectDescriptionSet> <xsl:if test="(not(../onix:NumberOfIllustrations)) and (not(onix:Number))"> <xsl:for-each select="onix:AncillaryContentDescription[(not(../../onix:NumberOfIllustrations)) and (not(../onix:Number))]"> <lido:descriptiveNoteValue> <xsl:value-of select="."/> </lido:descriptiveNoteValue> </xsl:for-each> </xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="(not(../onix:NumberOfIllustrations)) and (not(onix:Number))">`
		`<xsl:for-each select="../onix:IllustrationsNote[(not(../onix:NumberOfIllustrations)) and (not(../onix:AncillaryContent/onix:Number))]">`
		`<lido:descriptiveNoteValue>`
		`<xsl:attribute name="lido:label">Illustrations and other contents note</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:descriptiveNoteValue>`
		`</xsl:for-each>`
		`</xsl:if>`
		`</lido:objectDescriptionSet>`
		`</xsl:if>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:CollateralDetail/onix:TextContent">`
		`<xsl:if test="((onix:ContentAudience = '00') or (onix:ContentAudience = '03') or (onix:ContentAudience = '06')) and ((onix:Text/@textformat = '00') or (onix:Text/@textformat = '06') or (onix:Text/@textformat = '07')) and ((onix:TextType = '02') or (onix:TextType = '03') or (onix:TextType = '10') or (onix:TextType = '11') or (onix:TextType = '12') or (onix:TextType = '13'))">`
		`<lido:objectDescriptionSet>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                  <xsl:attribute name="lido:type">
                    <xsl:for-each select="onix:TextType">
                      <xsl:if test="position() = 1">
                        <xsl:variable name="idx63" select="index-of($map62/map,
normalize-space())"/>
                        <xsl:choose>
                          <xsl:when test="$idx63 > 0">
                            <xsl:value-of select="$map62/map[$idx63]/@value"/>
                          </xsl:when>
                          <xsl:otherwise>
                            <xsl:value-of select="."/>
                          </xsl:otherwise>
                        </xsl:choose>
                      </xsl:if>
                    </xsl:for-each>
                  </xsl:attribute>
```

LIDO section	LIDO subsection	XSLT	Comments
		see code below (row 1)	
		see code below (row 2)	

```
                  <xsl:for-each select="onix:Text">
                    <lido:descriptiveNoteValue>
                      <xsl:value-of select="."/>
                    </lido:descriptiveNoteValue>
                  </xsl:for-each>
                  <xsl:for-each select="onix:SourceTitle">
                    <lido:sourceDescriptiveNote>
                      <xsl:value-of select="."/>
                    </lido:sourceDescriptiveNote>
                  </xsl:for-each>
                </lido:objectDescriptionSet>
              </xsl:if>
            </xsl:for-each>
          </lido:objectDescriptionWrap>
```

```
          <lido:objectMeasurementsWrap>
            <xsl:for-each
select="onix:DescriptiveDetail/onix:AncillaryContent/onix:Number">
              <xsl:if test="(.)">
                <lido:objectMeasurementsSet>
                  <xsl:if test="(../onix:AncillaryContentDescription)">
                    <xsl:for-each select="../onix:AncillaryContentDescription[(.)]">
                      <lido:displayObjectMeasurements>
                        <xsl:attribute name="lido:label">Ancillary content
description</xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:displayObjectMeasurements>
                    </xsl:for-each>
                  </xsl:if>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:objectMeasurements>`
		`<lido:measurementsSet>`
		`<lido:measurementType>`
		`<xsl:attribute name="lido:label">Ancillary content type</xsl:attribute>Number of illustrations or other content items</lido:measurementType>`
		`<lido:measurementUnit>(count)</lido:measurementUnit>`
		`<xsl:for-each select=".">`
		`<xsl:if test="position() = 1">`
		`<lido:measurementValue>`
		`<xsl:value-of select="."/>`
		`</lido:measurementValue>`
		`</xsl:if>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                    </lido:measurementsSet>
                    <xsl:for-each select="../onix:AncillaryContentType">
                      <xsl:variable name="idx65" select="index-of($map64/map,
normalize-space())"/>
                      <xsl:choose>
                        <xsl:when test="$idx65 > 0">
                          <lido:extentMeasurements>
                            <xsl:value-of select="$map64/map[$idx65]/@value"/>
                          </lido:extentMeasurements>
                        </xsl:when>
                        <xsl:otherwise>
                          <lido:extentMeasurements>
                            <xsl:attribute name="lido:label">Ancillary content
type</xsl:attribute>
                            <xsl:value-of select="."/>
                          </lido:extentMeasurements>
                        </xsl:otherwise></xsl:choose></xsl:for-each>
                  </lido:objectMeasurements>
                </lido:objectMeasurementsSet>
              </xsl:if></xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:DescriptiveDetail/onix:NumberOfIllustrations">` `<xsl:if test="(.)">` `<lido:objectMeasurementsSet>` `<xsl:for-each select="../onix:IllustrationsNote">` `<lido:displayObjectMeasurements>` `<xsl:value-of select="."/>` `</lido:displayObjectMeasurements>` `</xsl:for-each>` `<lido:objectMeasurements>`	
		`<lido:measurementsSet>` `<lido:measurementType>Number of illustrations</lido:measurementType>` `<lido:measurementUnit>(count)</lido:measurementUnit>` `<xsl:for-each select=".">` `<xsl:if test="position() = 1">` `<lido:measurementValue>` `<xsl:value-of select="."/>` `</lido:measurementValue>` `</xsl:if>` `</xsl:for-each>` `</lido:measurementsSet>` `</lido:objectMeasurements>` `</lido:objectMeasurementsSet>` `</xsl:if>` `</xsl:for-each>`	

```
            <xsl:for-each
select="onix:DescriptiveDetail/onix:NumberOfIllustrations">
              <xsl:if test="(.)">
                <lido:objectMeasurementsSet>
                  <xsl:for-each select="../onix:IllustrationsNote">
                    <lido:displayObjectMeasurements>
                      <xsl:value-of select="."/>
                    </lido:displayObjectMeasurements>
                  </xsl:for-each>
                  <lido:objectMeasurements>
```

```
                    <lido:measurementsSet>
                      <lido:measurementType>Number of
illustrations</lido:measurementType>
                      <lido:measurementUnit>(count)</lido:measurementUnit>
                      <xsl:for-each select=".">
                        <xsl:if test="position() = 1">
                          <lido:measurementValue>
                            <xsl:value-of select="."/>
                          </lido:measurementValue>
                        </xsl:if>
                      </xsl:for-each>
                    </lido:measurementsSet>
                  </lido:objectMeasurements>
                </lido:objectMeasurementsSet>
              </xsl:if>
            </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:MapScale">`
		`  <xsl:if test="(.)">`
		`    <lido:objectMeasurementsSet>`
		
		`      <lido:displayObjectMeasurements>Map scale:Â 1:<xsl:for-each select=".">`
		`          <xsl:value-of select="."/>`
		`        </xsl:for-each>`
		`      </lido:displayObjectMeasurements>`	

		`      <lido:objectMeasurements>`
		`        <lido:measurementsSet>`
		`          <lido:measurementType>Map scale</lido:measurementType>`
		`          <lido:measurementUnit>1</lido:measurementUnit>`
		`          <xsl:for-each select=".">`
		`            <xsl:if test="position() = 1">`
		`              <lido:measurementValue>`
		`                <xsl:value-of select="."/>`
		`              </lido:measurementValue>`
		`            </xsl:if>`
		`          </xsl:for-each>`
		`        </lido:measurementsSet>`
		`      </lido:objectMeasurements>`
		`    </lido:objectMeasurementsSet>`
		`  </xsl:if>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:DescriptiveDetail/onix:Measure">` `<lido:objectMeasurementsSet>` `<lido:displayObjectMeasurements>` `<xsl:for-each select="onix:MeasureType">` `<xsl:variable name="idx67" select="index-of($map66/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx67 > 0">` `<xsl:value-of select="$map66/map[$idx67]/@value"/>` `</xsl:when>` `<xsl:otherwise>` `<xsl:value-of select="."/>` `</xsl:otherwise>` `</xsl:choose>` `</xsl:for-each>:Â <xsl:for-each select="onix:Measurement">` `<xsl:value-of select="."/>` `</xsl:for-each>Â <xsl:for-each select="onix:MeasureUnitCode">` `<xsl:value-of select="."/>` `</xsl:for-each>                </lido:displayObjectMeasurements>`	

```
            <xsl:for-each select="onix:DescriptiveDetail/onix:Measure">
              <lido:objectMeasurementsSet>
                <lido:displayObjectMeasurements>
                  <xsl:for-each select="onix:MeasureType">
                    <xsl:variable name="idx67" select="index-of($map66/map,
normalize-space())"/>
                    <xsl:choose>
                      <xsl:when test="$idx67 > 0">
                        <xsl:value-of select="$map66/map[$idx67]/@value"/>
                      </xsl:when>
                      <xsl:otherwise>
                        <xsl:value-of select="."/>
                      </xsl:otherwise>
                    </xsl:choose>
                  </xsl:for-each>:Â <xsl:for-each select="onix:Measurement">
                    <xsl:value-of select="."/>
                  </xsl:for-each>Â <xsl:for-each select="onix:MeasureUnitCode">
                    <xsl:value-of select="."/>
                  </xsl:for-each>                </lido:displayObjectMeasurements>
```

LIDO section	LIDO subsection	XSLT	Comments

```
                <lido:objectMeasurements>
                  <lido:measurementsSet>
                    <xsl:for-each select="onix:MeasureType">
                      <xsl:variable name="idx69" select="index-of($map68/map,
normalize-space())"/>
                      <xsl:choose>
                        <xsl:when test="$idx69 > 0">
                          <lido:measurementType>
                            <xsl:value-of select="$map68/map[$idx69]/@value"/>
                          </lido:measurementType>
                        </xsl:when>
                        <xsl:otherwise>
                          <lido:measurementType>
                            <xsl:attribute name="xml:lang">en</xsl:attribute>
                            <xsl:value-of select="."/>
                          </lido:measurementType>
                        </xsl:otherwise>
                      </xsl:choose>
                    </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`          <xsl:for-each select="onix:MeasureUnitCode">`
		`           <xsl:variable name="idx71" select="index-of($map70/map,`
		`normalize-space())"/>`
		`           <xsl:choose>`
		`            <xsl:when test="$idx71 > 0">`
		`             <lido:measurementUnit>`
		`              <xsl:value-of select="$map70/map[$idx71]/@value"/>`
		`             </lido:measurementUnit>`
		`            </xsl:when>`
		`            <xsl:otherwise>`
		`             <lido:measurementUnit>`
		`              <xsl:value-of select="."/>`
		`             </lido:measurementUnit>`
		`            </xsl:otherwise>`
		`           </xsl:choose>`
		`          </xsl:for-each>`	

		`          <xsl:for-each select="onix:Measurement">`
		`           <xsl:if test="position() = 1">`
		`            <lido:measurementValue>`
		`             <xsl:value-of select="."/>`
		`            </lido:measurementValue>`
		`           </xsl:if>`
		`          </xsl:for-each>`
		`         </lido:measurementsSet>`
		`        </lido:objectMeasurements>`
		`       </lido:objectMeasurementsSet>`
		`      </xsl:for-each>`
		`     </lido:objectMeasurementsWrap>`
		`    </lido:objectIdentificationWrap>`	

LIDO section	LIDO subsection	XSLT	Comments
Event	Event	`<lido:eventWrap>` `<lido:eventSet>` `<lido:event>` `<lido:eventType>` `<lido:conceptID>` `<xsl:attribute name="lido:type">URI</xsl:attribute>http://terminology.lido-schema.org/lido00012</lido:conceptID>` `<lido:term>` `<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>Creation</lido:term>` `</lido:eventType>` **`<xsl:for-each select="onix:DescriptiveDetail/onix:Contributor">`**	The LIDO creation event is structurally mapped to the ONIX contributor composite, since this is the defining feature of the event in ONIX, lacking an explicit event structure because the publication is the primary event of interest for a book product.
	Actor	`<lido:eventActor>` `<xsl:attribute name="lido:sortorder">` `<xsl:for-each select="onix:SequenceNumber">` `<xsl:if test="position() = 1">` `<xsl:value-of select="."/>` `</xsl:if></xsl:for-each></xsl:attribute>`	Maps the ONIX sequence number of contributors to the equivalent LIDO sort order. In this case, the numbering is really equivalent.

LIDO section	LIDO subsection	XSLT	Comments
	Actor	**<xsl:if test="(onix:ProfessionalAffiliation)">** <lido:displayActorInRole> <xsl:attribute name="lido:label">Professional position - affiliation</xsl:attribute> <xsl:for-each select="onix:ProfessionalAffiliation/onix:ProfessionalPosition"> <xsl:value-of select="."/> **</xsl:for-each>Â -Â <xsl:for-each select="onix:ProfessionalAffiliation/onix:Affiliation">** <xsl:value-of select="."/> </xsl:for-each> </lido:displayActorInRole> **</xsl:if>**	This portion concatenates the position and affiliation elements, separated by 2 hard space characters (ALT+0160) surrounding a hyphen – shown here as “Â –Â” – displaying exactly as in the @label describing the content so that end-users can make sense of the data. The “xsl:if” clause tests for the existence of the containing composite in the ONIX source, since no structural mapping can be made from 2 elements, and otherwise a blank lido:displayActorInRole could be generated.
		<xsl:for-each select="onix:ContributorDescription"> <lido:displayActorInRole> <xsl:attribute name="lido:label">Contributor description</xsl:attribute> <xsl:value-of select="."/> </lido:displayActorInRole> </xsl:for-each>	

LIDO section	LIDO subsection	XSLT	Comments
		```
		                  <lido:actorInRole>
		                    <lido:actor>
		                      <xsl:for-each
		select="onix:AlternativeName/onix:NameIdentifier/onix:IDValue">
		                        <lido:actorID>
		                          <xsl:attribute name="lido:pref">alternate</xsl:attribute>
		                          <xsl:attribute name="lido:type">
		                            <xsl:for-each select="../onix:NameIDType">
		                              <xsl:if test="position() = 1">
		                                <xsl:variable name="idx73" select="index-
		of($map72/map, normalize-space())"/>
		                                <xsl:choose>
		                                  <xsl:when test="$idx73 > 0">
		                                    <xsl:value-of
		select="$map72/map[$idx73]/@value"/>
		                                  </xsl:when>
		                                  <xsl:otherwise>
		                                    <xsl:value-of select="."/>
		                                  </xsl:otherwise><xsl:choose></xsl:if></xsl:for-
		each>
		                          </xsl:attribute>
		```

LIDO section	LIDO subsection	XSLT	Comments

```
    <xsl:if test="(../onix:IDTypeName)">
      <xsl:attribute name="lido:label">
        <xsl:for-each select="../onix:IDTypeName">
          <xsl:if test="position() = 1">
            <xsl:value-of select="."/>
          </xsl:if>
        </xsl:for-each>
      </xsl:attribute>
    </xsl:if>
    <xsl:value-of select="."/>
  </lido:actorID>
</xsl:for-each>
<xsl:for-each select="onix:NameIdentifier/onix:IDValue">
```

LIDO section	LIDO subsection	XSLT	Comments

		`  <lido:actorID>`
		`    <xsl:attribute name="lido:pref">preferred</xsl:attribute>`
		`    <xsl:attribute name="lido:type">`
		`      <xsl:for-each select="../onix:NameIDType">`
		`        <xsl:if test="position() = 1">`
		`          <xsl:value-of select="."/>`
		`        </xsl:if>`
		`      </xsl:for-each>`
		`    </xsl:attribute>`
		`    <xsl:if test="(../onix:IDTypeName)">`
		`      <xsl:attribute name="lido:label">`
		`        <xsl:for-each select="../onix:IDTypeName">`
		`          <xsl:if test="position() = 1">`
		`            <xsl:value-of select="."/>`
		`          </xsl:if></xsl:for-each></xsl:attribute></xsl:if>`
		`    <xsl:value-of select="."/>`
		`  </lido:actorID>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:AlternativeName">` `<lido:nameActorSet>` `<xsl:for-each select="onix:PersonName">` `<lido:appellationValue>` `<xsl:attribute name="lido:label">Person nameÂ -Â <xsl:for-each select="../onix:NameType">` `<xsl:if test="position() = 1">` `<xsl:variable name="idx75" select="index-of($map74/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx75 > 0">` `<xsl:value-of select="$map74/map[$idx75]/@value"/>` `</xsl:when><xsl:otherwise>` `<xsl:value-of select="."/>` `</xsl:otherwise></xsl:choose>` `</xsl:if></xsl:for-each>` `</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:appellationValue><xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="(not(onix:PersonName))">` `<xsl:for-each select="onix:PersonNameInverted[(not(../onix:PersonName))]">` `<lido:appellationValue>` `<xsl:attribute name="lido:label">Person name invertedÂ - Â <xsl:for-each select="../onix:NameType">` `<xsl:if test="position() = 1">` `<xsl:variable name="idx77" select="index-of($map76/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx77 > 0">` `<xsl:value-of select="$map76/map[$idx77]/@value"/>` `</xsl:when><xsl:otherwise>` `<xsl:value-of select="."/>` `</xsl:otherwise></xsl:choose>` `</xsl:if></xsl:for-each>` `</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:appellationValue></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                          <xsl:for-each select="onix:CorporateName">
                            <lido:appellationValue>
                              <xsl:attribute
                                  name="lido:label">Corporate nameÂ -Â <xsl:for-each
select="../onix:NameType">
                                  <xsl:if test="position() = 1">
                                    <xsl:variable name="idx79" select="index-
of($map78/map, normalize-space())"/>
                                    <xsl:choose>
                                    <xsl:when test="$idx79 > 0">
                                    <xsl:value-of
select="$map78/map[$idx79]/@value"/>
                                    </xsl:when>
                                    <xsl:otherwise>
                                    <xsl:value-of select="."/>
                                    </xsl:otherwise></xsl:choose></xsl:if>
                                </xsl:for-each>
                              </xsl:attribute>
                              <xsl:value-of select="."/>
                            </lido:appellationValue>
                          </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="(not(onix:CorporateName))">` `<xsl:for-each select="onix:CorporateNameInverted[(not(../onix:CorporateName))]">` `<lido:appellationValue>` `<xsl:attribute name="lido:label">Corporate name invertedÂ - Â <xsl:for-each select="../onix:NameType">` `<xsl:if test="position() = 1">` `<xsl:variable name="idx81" select="index-of($map80/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx81 > 0">` `<xsl:value-of select="$map80/map[$idx81]/@value"/>` `</xsl:when>` `<xsl:otherwise>` `<xsl:value-of select="."/>` `</xsl:otherwise></xsl:choose>` `</xsl:if></xsl:for-each></xsl:attribute>` `<xsl:value-of select="."/>` `</lido:appellationValue></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		<xsl:for-each select="onix:TitlesBeforeNames"> <lido:appellationValue> <xsl:attribute name="lido:label">Person name part 1: titles before namesÂ -Â <xsl:for-each select="../onix:NameType"> <xsl:if test="position() = 1"> <xsl:variable name="idx83" select="index-of($map82/map, normalize-space())"/> <xsl:choose> <xsl:when test="$idx83 > 0"> <xsl:value-of select="$map82/map[$idx83]/@value"/> </xsl:when> <xsl:otherwise> <xsl:value-of select="."/> </xsl:otherwise></xsl:choose></xsl:if></xsl:for-each> </xsl:attribute> <xsl:value-of select="."/> </lido:appellationValue> </xsl:for-each>	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:NamesBeforeKey">`
		`  <lido:appellationValue>`
		`    <xsl:attribute`
		`        name="lido:label">Person name part 2: names before key namesÂ -Â <xsl:for-each select="../onix:NameType">`
		`        <xsl:if test="position() = 1">`
		`          <xsl:variable name="idx85" select="index-of($map84/map, normalize-space())"/>`
		`          <xsl:choose>`
		`          <xsl:when test="$idx85 > 0">`
		`          <xsl:value-of select="$map84/map[$idx85]/@value"/>`
		`          </xsl:when>`
		`          <xsl:otherwise>`
		`          <xsl:value-of select="."/>`
		`          </xsl:otherwise></xsl:choose></xsl:if>`
		`      </xsl:for-each>`
		`    </xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		```
		                          <xsl:for-each select="onix:PrefixToKey">
		                            <lido:appellationValue>
		                              <xsl:attribute
		                                  name="lido:label">Person name part 3: prefix to
		key namesÂ -Â <xsl:for-each select="../onix:NameType">
		                                  <xsl:if test="position() = 1">
		                                    <xsl:variable name="idx87" select="index-
		of($map86/map, normalize-space())"/>
		                                    <xsl:choose>
		                                    <xsl:when test="$idx87 > 0">
		                                    <xsl:value-of
		select="$map86/map[$idx87]/@value"/>
		                                    </xsl:when>
		                                    <xsl:otherwise>
		                                    <xsl:value-of select="."/>
		                                    </xsl:otherwise></xsl:choose></xsl:if>
		                                </xsl:for-each>
		                              </xsl:attribute>
		                              <xsl:value-of select="."/>
		                            </lido:appellationValue>
		                          </xsl:for-each>
		```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:KeyNames">`
		`  <lido:appellationValue>`
		`    <xsl:attribute`
		`        name="lido:label">Person name part 4: key namesÂ -`
		`Â <xsl:for-each select="../onix:NameType">`
		`        <xsl:if test="position() = 1">`
		`          <xsl:variable name="idx89" select="index-`
		`of($map88/map, normalize-space())"/>`
		`          <xsl:choose>`
		`          <xsl:when test="$idx89 > 0">`
		`          <xsl:value-of`
		`select="$map88/map[$idx89]/@value"/>`
		`          </xsl:when>`
		`          <xsl:otherwise>`
		`          <xsl:value-of select="."/>`
		`          </xsl:otherwise></xsl:choose></xsl:if></xsl:for-`
		`each>`
		`    </xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:NamesAfterKey">`
		`<lido:appellationValue>`
		`<xsl:attribute`
		`name="lido:label">Person name part 5: names after key namesÂ -Â <xsl:for-each select="../onix:NameType">`
		`<xsl:if test="position() = 1">`
		`<xsl:variable name="idx91" select="index-of($map90/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx91 > 0">`
		`<xsl:value-of select="$map90/map[$idx91]/@value"/>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<xsl:value-of select="."/>`
		`</xsl:otherwise></xsl:choose></xsl:if>`
		`</xsl:for-each>`
		`</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:SuffixToKey">`
		`<lido:appellationValue>`
		`<xsl:attribute`
		`name="lido:label">Person name part 6: suffix after key namesÂ -Â <xsl:for-each select="../onix:NameType">`
		`<xsl:if test="position() = 1">`
		`<xsl:variable name="idx93" select="index-of($map92/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx93 > 0">`
		`<xsl:value-of select="$map92/map[$idx93]/@value"/>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<xsl:value-of select="."/>`
		`</xsl:otherwise></xsl:choose></xsl:if>`
		`</xsl:for-each>`
		`</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		(code below)	

```
                          <xsl:for-each select="onix:LettersAfterNames">
                            <lido:appellationValue>
                              <xsl:attribute
                                  name="lido:label">Person name part 7:
qualifications and honors after namesÂ -Â <xsl:for-each select="../onix:NameType">
                                  <xsl:if test="position() = 1">
                                    <xsl:variable name="idx95" select="index-
of($map94/map, normalize-space())"/>
                                    <xsl:choose>
                                    <xsl:when test="$idx95 > 0">
                                    <xsl:value-of
select="$map94/map[$idx95]/@value"/>
                                    </xsl:when>
                                    <xsl:otherwise>
                                    <xsl:value-of select="."/>
                                    </xsl:otherwise></xsl:choose></xsl:if>
                                </xsl:for-each>
                              </xsl:attribute>
                              <xsl:value-of select="."/>
                            </lido:appellationValue>
                          </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:TitlesAfterNames">`
		`<lido:appellationValue>`
		`<xsl:attribute`
		`name="lido:label">Person name part 8: titles after namesÂ -Â <xsl:for-each select="../onix:NameType">`
		`<xsl:if test="position() = 1">`
		`<xsl:value-of select="."/>`
		`</xsl:if>`
		`</xsl:for-each>`
		`</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`
		`</lido:nameActorSet>`
		`</xsl:for-each>`	

		`<lido:nameActorSet>`
		`<xsl:for-each select="onix:TitlesBeforeNames">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Person name part 1: titles before names</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:NamesBeforeKey">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Person name part 2: names before key names</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:PrefixToKey">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Person name part 3: prefix to key names</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:KeyNames">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Person name part 4: key names</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:NamesAfterKey">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Person name part 5:names after key names</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:SuffixToKey">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Person name part 6: suffix after key names</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:LettersAfterNames">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Person name part 7: qualifications and honors after names</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:TitlesAfterNames">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Person name part 8: titles after names</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:CorporateName">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Corporate contributor name</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="(not(onix:CorporateName))"> <xsl:for-each select="onix:CorporateNameInverted[(not(../onix:CorporateName))]"> <lido:appellationValue> <xsl:attribute name="lido:label">Corporate contributor name, inverted</xsl:attribute> <xsl:value-of select="."/> </lido:appellationValue> </xsl:for-each> </xsl:if>`	
		`<xsl:if test="(not(onix:PersonName))"> <xsl:for-each select="onix:PersonNameInverted[(not(../onix:PersonName))]"> <lido:appellationValue> <xsl:attribute name="lido:label">Person name, inverted</xsl:attribute> <xsl:value-of select="."/> </lido:appellationValue> </xsl:for-each> </xsl:if>`	
		`<xsl:for-each select="onix:PersonName"> <lido:appellationValue> <xsl:attribute name="lido:label">Person name</xsl:attribute> <xsl:value-of select="."/> </lido:appellationValue> </xsl:for-each> </lido:nameActorSet>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:ContributorPlace">`
		`<lido:nationalityActor>`
		`<xsl:for-each select="onix:RegionCode">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>`
		`<xsl:attribute name="lido:label">`
		`<xsl:for-each select="../onix:ContributorPlaceRelator">`
		`<xsl:if test="position() = 1">`
		`<xsl:variable name="idx97" select="index-of($map96/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx97 > 0">`
		`<xsl:value-of select="$map96/map[$idx97]/@value"/>`
		`</xsl:when><xsl:otherwise>`
		`<xsl:value-of select="."/>`
		`</xsl:otherwise></xsl:choose></xsl:if>`
		`</xsl:for-each>Â region code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID></xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:CountryCode">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">ISO 3166-1</xsl:attribute>`
		`<xsl:attribute name="lido:label">`
		`<xsl:for-each select="../onix:ContributorPlaceRelator">`
		`<xsl:if test="position() = 1">`
		`<xsl:variable name="idx99" select="index-of($map98/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx99 > 0">`
		`<xsl:value-of select="$map98/map[$idx99]/@value"/>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<xsl:value-of select="."/>`
		`</xsl:otherwise></xsl:choose></xsl:if>`
		`</xsl:for-each>Â country code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID></xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                          <xsl:for-each select="onix:RegionCode">
                            <xsl:variable name="idx101" select="index-
of($map100/map, normalize-space())"/>
                            <xsl:choose>
                              <xsl:when test="$idx101 &gt; 0">
                                <lido:term>
                                  <xsl:value-of
select="$map100/map[$idx101]/@value"/>
                                </lido:term>
                              </xsl:when>
                              <xsl:otherwise>
                                <lido:term>
                                  <xsl:attribute
name="lido:addedSearchTerm">no</xsl:attribute>
                                  <xsl:attribute name="lido:label">
                                    <xsl:for-each
select="../onix:ContributorPlaceRelator">
                                    <xsl:if test="position() = 1">
                                    <xsl:value-of select="."/>
                                    </xsl:if>
                                    </xsl:for-each> region</xsl:attribute>
                                  <xsl:value-of select="."/>
                                </lido:term>
                              </xsl:otherwise>
                            </xsl:choose>
                          </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:CountryCode">`
		`<xsl:variable name="idx103" select="index-of($map102/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx103 &gt; 0">`
		`<lido:term>`
		`<xsl:value-of select="$map102/map[$idx103]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`<xsl:attribute name="lido:label">`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="../onix:ContributorPlaceRelator">`
		`<xsl:if test="position() = 1">`
		`<xsl:variable name="idx105" select="index-of($map104/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx105 &gt; 0">`
		`<xsl:value-of select="$map104/map[$idx105]/@value"/>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<xsl:value-of select="."/>`
		`</xsl:otherwise>`
		`</xsl:choose>`
		`</xsl:if>`
		`</xsl:for-each> country</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term></xsl:otherwise></xsl:choose>`
		`</xsl:for-each>`
		`</lido:nationalityActor>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:vitalDatesActor>`
		`<xsl:if test="onix:ContributorDate/onix:ContributorDateRole = '50'">`
		`<xsl:for-each select="onix:ContributorDate/onix:Date[../onix:ContributorDateRole = '50']">`
		`<xsl:if test="position() = 1">`
		`<lido:earliestDate>`
		`<xsl:attribute name="lido:type">`
		`<xsl:for-each select="@dateformat">`
		`<xsl:if test="position() = 1">`
		`<xsl:value-of select="."/>`
		`</xsl:if>`
		`</xsl:for-each>`
		`</xsl:attribute>`
		`<xsl:attribute name="lido:label">Date of birth</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:earliestDate>`
		`</xsl:if></xsl:for-each>`
		`</xsl:if>`
			There is a small bug in the XSLT here, since MINT does not allow a conditional statement for, or a structural mapping to lido:vitalDatesActor and hence if no onix:ContributorDate is present, empty elements may be generated for lido:VitalDatesActor.

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="onix:ContributorDate/onix:ContributorDateRole = '51'">` `<xsl:for-each select="onix:ContributorDate/onix:Date[../onix:ContributorDateRole = '51']">` `<xsl:if test="position() = 1">` `<lido:latestDate>` `<xsl:attribute name="lido:type">` `<xsl:for-each select="@dateformat">` `<xsl:if test="position() = 1">` `<xsl:value-of select="."/>` `</xsl:if>` `</xsl:for-each>` `</xsl:attribute>` `<xsl:attribute name="lido:label">Date of death</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:latestDate>` `</xsl:if></xsl:for-each></xsl:if>` `</lido:vitalDatesActor></lido:actor>`	
		`<lido:roleActor>` `<xsl:for-each select="onix:ContributorRole">` `<lido:conceptID>` `<xsl:attribute name="lido:type">Local</xsl:attribute>` `<xsl:attribute name="lido:label">Contributor role code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:conceptID>` `</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                      <xsl:for-each select="onix:ContributorRole">
                        <xsl:variable name="idx107" select="index-of($map106/map,
normalize-space())"/>
                        <xsl:choose>
                          <xsl:when test="$idx107 &gt; 0">
                            <lido:term>
                              <xsl:value-of select="$map106/map[$idx107]/@value"/>
                            </lido:term>
                          </xsl:when>
                          <xsl:otherwise>
                            <lido:term>
                              <xsl:attribute
name="lido:addedSearchTerm">no</xsl:attribute>
                              <xsl:attribute name="lido:label">Contributor
role</xsl:attribute>
                              <xsl:value-of select="."/>
                            </lido:term>
                          </xsl:otherwise></xsl:choose></xsl:for-each>
                    </lido:roleActor></lido:actorInRole></lido:eventActor>
</xsl:for-each></lido:event></lido:eventSet>
```

LIDO section	LIDO subsection	XSLT	Comments

```
          <xsl:for-each select="onix:CollateralDetail/onix:TextContent">
            <xsl:if test="((onix:ContentAudience = '00') or (onix:ContentAudience =
'03') or (onix:ContentAudience = '06')) and ((onix:TextType = '06') or
(onix:TextType = '07') or (onix:TextType = '08'))">
              <lido:eventSet>
                <lido:event>
                  <lido:eventType>
                    <lido:conceptID>
                      <xsl:attribute
name="lido:type">Local</xsl:attribute>http://terminology.lido-
schema.org/lido00003</lido:conceptID>
                    <lido:term>
                      <xsl:attribute
name="lido:addedSearchTerm">no</xsl:attribute>(Non-specified)</lido:term>
                  </lido:eventType>
```

LIDO section	LIDO subsection	XSLT	Comments
		(code below)	

```
                  <xsl:for-each select="onix:TextSourceCorporate">
                    <lido:eventActor>
                      <lido:actorInRole>
                        <lido:actor>
                          <lido:nameActorSet>
                            <xsl:for-each select=".">
                              <lido:appellationValue>
                                <xsl:attribute name="lido:label">Corporate source of 
text</xsl:attribute>
                                <xsl:value-of select="."/>
                              </lido:appellationValue>
                            </xsl:for-each>
                          </lido:nameActorSet>
                        </lido:actor>
                      </lido:actorInRole>
                    </lido:eventActor>
                  </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments
		(see below, row 1)	
		(see below, row 2)	

```
<xsl:for-each select="onix:TextAuthor">
  <lido:eventActor>
    <lido:actorInRole>
      <lido:actor>
        <lido:nameActorSet>
          <xsl:for-each select=".">
            <lido:appellationValue>
              <xsl:attribute name="lido:label">Text
author</xsl:attribute>
              <xsl:value-of select="."/>
            </lido:appellationValue>
          </xsl:for-each>
        </lido:nameActorSet>
      </lido:actor>
    </lido:actorInRole>
  </lido:eventActor>
</xsl:for-each>
```

```
<lido:eventDate>
  <lido:date>
    <xsl:if test="onix:ContentDate/onix:ContentDateRole = '01'">
      <xsl:for-each
select="onix:ContentDate/onix:Date[../onix:ContentDateRole = '01']">
        <xsl:if test="position() = 1">
          <lido:earliestDate>
            <xsl:value-of select="."/>
          </lido:earliestDate>
        </xsl:if>
      </xsl:for-each>
    </xsl:if>
```

LIDO section	LIDO subsection	XSLT	Comments

```
                      <xsl:if test="onix:ContentDate/onix:ContentDateRole = '01'">
                        <xsl:for-each
select="onix:ContentDate/onix:Date[../onix:ContentDateRole = '01']">
                          <xsl:if test="position() = 1">
                            <lido:latestDate>
                              <xsl:value-of select="."/>
                            </lido:latestDate>
                          </xsl:if>
                        </xsl:for-each>
                      </xsl:if>
                    </lido:date>
                  </lido:eventDate>
```

```
                  <lido:eventDescriptionSet>
                    <xsl:attribute name="lido:type">
                      <xsl:for-each select="onix:TextType">
                        <xsl:if test="position() = 1">
                          <xsl:variable name="idx109" select="index-of($map108/map,
normalize-space())"/>
                          <xsl:choose>
                            <xsl:when test="$idx109 &gt; 0">
                              <xsl:value-of select="$map108/map[$idx109]/@value"/>
                            </xsl:when>
                            <xsl:otherwise>
                              <xsl:value-of select="."/>
                            </xsl:otherwise>
                          </xsl:choose>
                        </xsl:if>
                      </xsl:for-each>
                    </xsl:attribute>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:Text">`
		`<lido:descriptiveNoteValue>`
		`<xsl:value-of select="."/>`
		`</lido:descriptiveNoteValue>`
		`</xsl:for-each>`
		`<xsl:for-each select="onix:SourceTitle">`
		`<lido:sourceDescriptiveNote>`
		`<xsl:value-of select="."/>`
		`</lido:sourceDescriptiveNote>`
		`</xsl:for-each>`
		`</lido:eventDescriptionSet>`
		`</lido:event>`
		`</lido:eventSet>`
		`</xsl:if>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:CollateralDetail/onix:CitedContent">`
		`<xsl:if test="(onix:ContentAudience = '00') or (onix:ContentAudience = '03') or (onix:ContentAudience = '06')">`
		`<lido:eventSet>`
		`<lido:event>`
		`<lido:eventType>`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>http://terminology.lido-schema.org/lido00003</lido:conceptID>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>(Non-specified)</lido:term>`
		`</lido:eventType>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:eventDate>`
		`<lido:date>`
		`<xsl:if test="(onix:ContentDate/onix:ContentDateRole = '01') or (onix:ContentDate/onix:ContentDateRole = '04')">`
		`<xsl:for-each select="onix:ContentDate/onix:Date[(../onix:ContentDateRole = '01') or (../onix:ContentDateRole = '04')]">`
		`<xsl:if test="position() = 1">`
		`<lido:earliestDate>`
		`<xsl:attribute name="lido:type">`
		`<xsl:for-each select="@dateformat">`
		`<xsl:if test="position() = 1">`
		`<xsl:value-of select="."/>`
		`</xsl:if>`
		`</xsl:for-each>`
		`</xsl:attribute>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                              <xsl:attribute name="lido:label">
                                <xsl:for-each select="../onix:ContentDateRole">
                                  <xsl:if test="position() = 1">
                                    <xsl:variable name="idx111" select="index-
of($map110/map, normalize-space())"/>
                                    <xsl:choose>
                                    <xsl:when test="$idx111 &gt; 0">
                                    <xsl:value-of
select="$map110/map[$idx111]/@value"/>
                                    </xsl:when>
                                    <xsl:otherwise>
                                    <xsl:value-of select="."/>
                                    </xsl:otherwise>
                                    </xsl:choose>
                                  </xsl:if>
                                </xsl:for-each>
                              </xsl:attribute>
                              <xsl:value-of select="."/>
                            </lido:earliestDate>
                          </xsl:if></xsl:for-each></xsl:if>
```

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="(onix:ContentDate/onix:ContentDateRole = '01') or (onix:ContentDate/onix:ContentDateRole = '04')">` `<xsl:for-each select="onix:ContentDate/onix:Date[(../onix:ContentDateRole = '01') or (../onix:ContentDateRole = '04')]">` `<xsl:if test="position() = 1">` `<lido:latestDate>` `<xsl:attribute name="lido:type">` `<xsl:for-each select="@dateformat">` `<xsl:if test="position() = 1">` `<xsl:value-of select="."/>` `</xsl:if>` `</xsl:for-each>` `</xsl:attribute>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:attribute name="lido:label">`
		`<xsl:for-each select="../onix:ContentDateRole">`
		`<xsl:if test="position() = 1">`
		`<xsl:variable name="idx113" select="index-of($map112/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx113 &gt; 0">`
		`<xsl:value-of select="$map112/map[$idx113]/@value"/>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<xsl:value-of select="."/>`
		`</xsl:otherwise>`
		`</xsl:choose>`
		`</xsl:if>`
		`</xsl:for-each>`
		`</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:latestDate>`
		`</xsl:if></xsl:for-each></xsl:if></lido:date></lido:eventDate>`	

		`<lido:thingPresent>`
		`<lido:object>`
		`<xsl:for-each select="onix:ResourceLink">`
		`<lido:objectWebResource>`
		`<xsl:value-of select="."/>`
		`</lido:objectWebResource>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:CitedContentType">`
		`  <xsl:variable name="idx115" select="index-of($map114/map, normalize-space())"/>`
		`  <xsl:choose>`
		`    <xsl:when test="$idx115 &gt; 0">`
		`      <lido:objectNote>`
		`        <xsl:value-of select="$map114/map[$idx115]/@value"/>`
		`      </lido:objectNote>`
		`    </xsl:when>`
		`    <xsl:otherwise>`
		`      <lido:objectNote>`
		`        <xsl:attribute name="lido:label">Cited content type</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:objectNote>`
		`    </xsl:otherwise>`
		`  </xsl:choose>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:SourceType">`
		`  <lido:objectNote>`
		`    <xsl:attribute name="lido:label">Source type</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectNote>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:SourceTitle">` `<lido:objectNote>` `<xsl:attribute name="lido:label">Source title</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:objectNote>` `</xsl:for-each>`	
		`<xsl:for-each select="onix:PositionOnList">` `<lido:objectNote>` `<xsl:attribute name="lido:label">Position on list</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:objectNote>` `</xsl:for-each>`	
		`<xsl:for-each select="onix:ListName">` `<lido:objectNote>` `<xsl:attribute name="lido:label">Name of bestseller list</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:objectNote>` `</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:CitationNote">`
		`<lido:objectNote>`
		`<xsl:attribute name="lido:type">Citation note</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:objectNote>`
		`</xsl:for-each>`
		`</lido:object>`
		`</lido:thingPresent>`
		`</lido:event>`
		`</lido:eventSet>`
		`</xsl:if>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:DescriptiveDetail/onix:Prize">`
		`<lido:eventSet>`
		`<lido:event>`
		`<lido:eventType>`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>http://terminology.lido-schema.org/lido00003</lido:conceptID>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>(Non-specified)</lido:term>`
		`</lido:eventType>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:roleInEvent>`
		`<xsl:for-each select="onix:PrizeCode">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>`
		`<xsl:attribute name="lido:label">Prize or award achievement code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:PrizeCode">`
		`<xsl:variable name="idx117" select="index-of($map116/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx117 &gt; 0">`
		`<lido:term>`
		`<xsl:value-of select="$map116/map[$idx117]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`<xsl:attribute name="lido:label">Prize or award acheivement</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term></xsl:otherwise></xsl:choose>`
		`</xsl:for-each></lido:roleInEvent>`	

LIDO section	LIDO subsection	XSLT	Comments

```
        <lido:eventName>
          <xsl:for-each select="onix:PrizeName">
            <lido:appellationValue>
              <xsl:attribute name="xml:lang">
                <xsl:for-each select="@language">
                  <xsl:if test="position() = 1">
                    <xsl:value-of select="."/>
                  </xsl:if>
                </xsl:for-each>
              </xsl:attribute>
              <xsl:attribute name="lido:label">Prize or award
name</xsl:attribute>
              <xsl:value-of select="."/>
            </lido:appellationValue></xsl:for-each></lido:eventName>
```

```
        <lido:eventActor>
          <xsl:for-each select="onix:PrizeJury">
            <lido:displayActorInRole>
              <xsl:attribute name="xml:lang">
                <xsl:for-each select="@language">
                  <xsl:if test="position() = 1">
                    <xsl:value-of select="."/>
                  </xsl:if>
                </xsl:for-each>
              </xsl:attribute>
              <xsl:attribute name="lido:label">Prize jury</xsl:attribute>
              <xsl:value-of select="."/>
            </lido:displayActorInRole>
          </xsl:for-each>
        </lido:eventActor>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:eventDate>`
		`  <xsl:for-each select="onix:PrizeYear">`
		`    <lido:displayDate>`
		`      <xsl:attribute name="lido:label">Prize or award`
		`year</xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:displayDate>`
		`  </xsl:for-each>`
		`  <lido:date>`
		`    <xsl:for-each select="onix:PrizeYear">`
		`      <xsl:if test="position() = 1">`
		`        <lido:earliestDate>`
		`          <xsl:attribute name="lido:type">YYYY</xsl:attribute>`
		`          <xsl:attribute name="lido:label">Prize or award`
		`year</xsl:attribute>`
		`          <xsl:value-of select="."/>`
		`        </lido:earliestDate>`
		`      </xsl:if>`
		`    </xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:PrizeYear">`
		`<xsl:if test="position() = 1">`
		`<lido:latestDate>`
		`<xsl:attribute name="lido:type">YYYY</xsl:attribute>`
		`<xsl:attribute name="lido:label">Prize or award year</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:latestDate>`
		`</xsl:if>`
		`</xsl:for-each>`
		`</lido:date>`
		`</lido:eventDate>`	

		`<lido:eventPlace>`
		`<lido:place>`
		`<xsl:for-each select="onix:PrizeCountry">`
		`<lido:placeID>`
		`<xsl:attribute name="lido:type">ISO 3166-1</xsl:attribute>`
		`<xsl:attribute name="lido:label">Prize or award country code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:placeID>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		(code below, row 1)	
		(code below, row 2)	

```
                    <lido:namePlaceSet>
                      <xsl:for-each select="onix:PrizeCountry">
                        <xsl:variable name="idx119" select="index-of($map118/map, normalize-space())"/>
                        <xsl:choose>
                          <xsl:when test="$idx119 &gt; 0">
                            <lido:appellationValue>
                              <xsl:value-of select="$map118/map[$idx119]/@value"/>
                            </lido:appellationValue>
                          </xsl:when>
```

```
                          <xsl:otherwise>
                            <lido:appellationValue>
                              <xsl:attribute name="lido:label">Prize or award country</xsl:attribute>
                              <xsl:value-of select="."/>
                            </lido:appellationValue>
                          </xsl:otherwise></xsl:choose></xsl:for-each>
                    </lido:namePlaceSet></lido:place>
                </lido:eventPlace></lido:event></lido:eventSet>
          </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:PublishingDetail/onix:Publisher">`
		`<xsl:if test="(onix:PublishingRole = '01') or (onix:PublishingRole = '02')">`
		`<lido:eventActor>`
		`<xsl:for-each select="onix:PublisherName">`
		`<lido:displayActorInRole>`
		`<xsl:value-of select="."/>`
		`</lido:displayActorInRole>`
		`</xsl:for-each>`
		`<lido:actorInRole>`
		`<lido:actor>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:PublisherIdentifier/onix:IDValue">` `<lido:actorID>` `<xsl:attribute name="lido:type">` `<xsl:for-each select="../onix:PublisherIDType">` `<xsl:if test="position() = 1">` `<xsl:variable name="idx121" select="index-of($map120/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx121 &gt; 0">` `<xsl:value-of select="$map120/map[$idx121]/@value"/>` `</xsl:when>` `<xsl:otherwise>` `<xsl:value-of select="."/>` `</xsl:otherwise></xsl:choose></xsl:if></xsl:for-each>` `</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:actorID>` `</xsl:for-each>`	
		`<lido:nameActorSet>` `<xsl:for-each select="onix:PublisherName">` `<lido:appellationValue>` `<xsl:value-of select="."/>` `</lido:appellationValue>` `</xsl:for-each>` `</lido:nameActorSet>` `</lido:actor>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:roleActor>`
		`<xsl:for-each select="onix:PublishingRole">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>`
		`<xsl:attribute name="lido:label">Publishing role code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:PublishingRole">`
		`<xsl:variable name="idx123" select="index-of($map122/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx123 &gt; 0">`
		`<lido:term>`
		`<xsl:value-of select="$map122/map[$idx123]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`<xsl:attribute name="lido:label">Publishing role</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term></xsl:otherwise></xsl:choose></xsl:for-each>`
		`</lido:roleActor></lido:actorInRole></lido:eventActor>`
		`</xsl:if></xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:PublishingDetail/onix:Imprint">`
		`  <lido:eventActor>`
		`    <xsl:for-each select="onix:ImprintName">`
		`      <lido:displayActorInRole>`
		`        <xsl:attribute name="lido:label">Imprint name</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:displayActorInRole>`
		`    </xsl:for-each>`	

		`<lido:actorInRole>`
		`  <lido:actor>`
		`    <xsl:for-each select="onix:ImprintIdentifier/onix:IDValue">`
		`      <lido:actorID>`
		`        <xsl:attribute name="lido:type">`
		`          <xsl:for-each select="../onix:ImprintIDType">`
		`            <xsl:if test="position() = 1">`
		`              <xsl:variable name="idx125" select="index-`
		`of($map124/map, normalize-space())"/>`
		`              <xsl:choose>`
		`                <xsl:when test="$idx125 &gt; 0">`
		`                  <xsl:value-of`
		`select="$map124/map[$idx125]/@value"/>`
		`                </xsl:when>`
		`                <xsl:otherwise>`
		`                  <xsl:value-of select="."/>`
		`                </xsl:otherwise></xsl:choose></xsl:if></xsl:for-`
		`each>`
		`        </xsl:attribute>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="(../onix:IDTypeName)">`
		`  <xsl:attribute name="lido:label">`
		`    <xsl:for-each select="../onix:IDTypeName">`
		`      <xsl:if test="position() = 1">`
		`        <xsl:value-of select="."/>`
		`      </xsl:if>`
		`    </xsl:for-each>`
		`  </xsl:attribute>`
		`</xsl:if>`
		`<xsl:value-of select="."/>`
		`</lido:actorID>`
		`</xsl:for-each>`	

		`<lido:nameActorSet>`
		`  <xsl:for-each select="onix:ImprintName">`
		`    <lido:appellationValue>`
		`      <xsl:attribute name="lido:label">Imprint name</xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:appellationValue>`
		`  </xsl:for-each>`
		`</lido:nameActorSet>`
		`</lido:actor>`
		`</lido:actorInRole>`
		`</lido:eventActor>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:eventDate>`
		`<lido:date>`
		`<xsl:if`
		`test="onix:PublishingDetail/onix:PublishingDate/onix:PublishingDateRole = '01'">`
		`<xsl:for-each`
		`select="onix:PublishingDetail/onix:PublishingDate/onix:Date[../onix:PublishingDateRo`
		`le = '01']">`
		`<xsl:if test="position() = 1">`
		`<lido:earliestDate>`
		`<xsl:attribute name="lido:type">`
		`<xsl:for-each select="@dateformat">`
		`<xsl:if test="position() = 1">`
		`<xsl:value-of select="."/>`
		`</xsl:if></xsl:for-each>`
		`</xsl:attribute>`
		`<xsl:attribute name="lido:label">Publication`
		`date</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:earliestDate>`
		`</xsl:if></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		<xsl:if test="onix:PublishingDetail/onix:PublishingDate/onix:PublishingDateRole = '01'"> <xsl:for-each select="onix:PublishingDetail/onix:PublishingDate/onix:Date[../onix:PublishingDateRole = '01']"> <xsl:if test="position() = 1"> <lido:latestDate> <xsl:attribute name="lido:type"> <xsl:for-each select="@dateformat"> <xsl:if test="position() = 1"> <xsl:value-of select="."/> </xsl:if> </xsl:for-each> </xsl:attribute> <xsl:attribute name="lido:label">Publication date</xsl:attribute> <xsl:value-of select="."/> </lido:latestDate> </xsl:if></xsl:for-each></xsl:if> </lido:date></lido:eventDate>	
		<xsl:for-each select="onix:PublishingDetail/onix:CountryOfPublication"> <lido:eventPlace> <lido:place> <xsl:for-each select="."> <lido:placeID> <xsl:attribute name="lido:type">ISO 3166-1</xsl:attribute> <xsl:value-of select="."/> </lido:placeID> </xsl:for-each>	

LIDO section	LIDO subsection	XSLT	Comments
		<lido:namePlaceSet>	
		<xsl:for-each select=".">	
		<xsl:variable name="idx127" select="index-of($map126/map,	
		normalize-space())"/>	
		<xsl:choose>	
		<xsl:when test="$idx127 &gt; 0">	
		<lido:appellationValue>	
		<xsl:value-of select="$map126/map[$idx127]/@value"/>	
		</lido:appellationValue>	
		</xsl:when>	
		<xsl:otherwise>	
		<lido:appellationValue>	
		<xsl:attribute name="lido:label">Country of	
		publication</xsl:attribute>	
		<xsl:value-of select="."/>	
		</lido:appellationValue>	
		</xsl:otherwise></xsl:choose></xsl:for-each>	
		</lido:namePlaceSet></lido:place></lido:eventPlace>	
		</xsl:for-each>	

LIDO section	LIDO subsection	XSLT	Comments

```
              <xsl:for-each select="onix:PublishingDetail/onix:CityOfPublication">
                <lido:eventPlace>
                  <xsl:for-each select=".">
                    <lido:displayPlace>
                      <xsl:value-of select="."/>
                    </lido:displayPlace>
                  </xsl:for-each>
                </lido:eventPlace>
              </xsl:for-each>
            </lido:event>
          </lido:eventSet>
        </lido:eventWrap>
```

```
        <lido:objectRelationWrap>
          <lido:subjectWrap>
            <xsl:for-each select="onix:DescriptiveDetail/onix:NameAsSubject">
              <lido:subjectSet>
                <xsl:if test="(../onix:Subject/onix:MainSubject)"/>
                <lido:subject>
                  <lido:subjectActor>
                    <xsl:if test="(not(onix:PersonName))">
                      <xsl:for-each
select="onix:PersonNameInverted[(not(../onix:PersonName))]">
                        <lido:displayActor>
                          <xsl:attribute name="lido:label">Person name
inverted</xsl:attribute>
                          <xsl:value-of select="."/>
                        </lido:displayActor>
                      </xsl:for-each>
                    </xsl:if>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:CorporateName">`
		`  <lido:displayActor>`
		`    <xsl:attribute name="lido:label">Corporate name</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:displayActor>`
		`</xsl:for-each>`
		`<xsl:if test="(not(onix:CorporateName))">`
		`  <xsl:for-each select="onix:CorporateNameInverted[(not(../onix:CorporateName))]">`
		`    <lido:displayActor>`
		`      <xsl:value-of select="."/>`
		`    </lido:displayActor>`
		`  </xsl:for-each>`
		`</xsl:if>`	

		`<xsl:for-each select="onix:PersonName">`
		`  <lido:displayActor>`
		`    <xsl:value-of select="."/>`
		`  </lido:displayActor>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<lido:actor>` `<xsl:for-each select="onix:NameIdentifier/onix:IDValue">` `<lido:actorID>` `<xsl:attribute name="lido:type">` `<xsl:for-each select="../onix:NameIDType">` `<xsl:if test="position() = 1">` `<xsl:variable name="idx129" select="index-of($map128/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx129 &gt; 0">` `<xsl:value-of select="$map128/map[$idx129]/@value"/>` `</xsl:when>` `<xsl:otherwise>` `<xsl:value-of select="."/>` `</xsl:otherwise></xsl:choose></xsl:if></xsl:for-each>` `</xsl:attribute>` `<xsl:attribute name="lido:label">Name identifier code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:actorID></xsl:for-each>`	
		`<lido:nameActorSet>` `<xsl:for-each select="onix:NamesBeforeKey">` `<lido:appellationValue>` `<xsl:attribute name="lido:label">Person name part 2: names before key names</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:appellationValue>` `</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:PrefixToKey">`
		`  <lido:appellationValue>`
		`    <xsl:attribute name="lido:label">Person name part 3: prefix to key names</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:KeyNames">`
		`  <lido:appellationValue>`
		`    <xsl:attribute name="lido:pref">preferred</xsl:attribute>`
		`    <xsl:attribute name="lido:label">Person name part 4: key names</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:NamesAfterKey">`
		`  <lido:appellationValue>`
		`    <xsl:attribute name="lido:label">Person name part 5: names after key names</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:LettersAfterNames">`
		`  <lido:appellationValue>`
		`    <xsl:attribute name="lido:label">Person name part 7: qualifications and honors after names</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:TitlesAfterNames">`
		`  <lido:appellationValue>`
		`    <xsl:attribute name="lido:label">Person name part 8: titles after names</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:SuffixToKey">`
		`  <lido:appellationValue>`
		`    <xsl:attribute name="lido:label">Person name part 6: suffix after key names</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:TitlesBeforeNames">`
		`  <lido:appellationValue>`
		`    <xsl:attribute name="lido:label">Person name part 1:titles before names</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:appellationValue>`
		`</xsl:for-each>`
		`</lido:nameActorSet></lido:actor></lido:subjectActor>`
		`</lido:subject></lido:subjectSet>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:DescriptiveDetail/onix:Subject">`
		`  <lido:subjectSet>`
		`    <xsl:if test="(onix:MainSubject)">`
		`      <xsl:attribute name="lido:sortorder">1</xsl:attribute>`
		`    </xsl:if>`
		`    <lido:subject>`
		`      <lido:subjectConcept>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                    <xsl:for-each select="onix:SubjectCode">
                      <lido:conceptID>
                        <xsl:attribute name="lido:type">
                          <xsl:for-each select="../onix:SubjectSchemeIdentifier">
                            <xsl:if test="position() = 1">
                              <xsl:variable name="idx131" select="index-
of($map130/map, normalize-space())"/>
                              <xsl:choose>
                                <xsl:when test="$idx131 &gt; 0">
                                  <xsl:value-of
select="$map130/map[$idx131]/@value"/>
                                </xsl:when>
                                <xsl:otherwise>
                                  <xsl:value-of select="."/>
                                </xsl:otherwise></xsl:choose></xsl:if></xsl:for-
each>
                        </xsl:attribute>
```

LIDO section	LIDO subsection	XSLT	Comments

```
                        <xsl:attribute name="lido:label">
                          <xsl:for-each select="../onix:SubjectSchemeIdentifier">
                            <xsl:if test="position() = 1">
                              <xsl:variable name="idx133" select="index-
of($map132/map, normalize-space())"/>
                              <xsl:choose>
                                <xsl:when test="$idx133 &gt; 0">
                                  <xsl:value-of
select="$map132/map[$idx133]/@value"/>
                                </xsl:when>
                                <xsl:otherwise>
                                  <xsl:value-of select="."/>
                                </xsl:otherwise></xsl:choose></xsl:if>
                          </xsl:for-each> version <xsl:for-each
select="../onix:SubjectSchemeVersion">
                            <xsl:if test="position() = 1">
                              <xsl:value-of select="."/>
                            </xsl:if></xsl:for-each></xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:conceptID></xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

```
                    <xsl:for-each select="onix:SubjectHeadingText">
                      <lido:term>
                        <xsl:attribute
name="lido:addedSearchTerm">no</xsl:attribute>
                        <xsl:attribute name="xml:lang">
                          <xsl:for-each select="@language">
                            <xsl:if test="position() = 1">
                              <xsl:value-of select="."/>
                            </xsl:if>
                          </xsl:for-each>
                        </xsl:attribute>
                        <xsl:attribute name="lido:label">Subject heading
text</xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:term>
                    </xsl:for-each>
                  </lido:subjectConcept></lido:subject></lido:subjectSet>
            </xsl:for-each>
          </lido:subjectWrap>
```

LIDO section	LIDO subsection	XSLT	Comments
		`<lido:relatedWorksWrap>` `<xsl:for-each select="onix:DescriptiveDetail/onix:Collection">` `<lido:relatedWorkSet>` `<xsl:if test="(../onix:ProductPart/onix:PrimaryPart)"/>` `<lido:relatedWork>` `<xsl:if test="(onix:TitleDetail/onix:TitleStatement)">` `<xsl:for-each select="onix:TitleDetail/onix:TitleStatement[(.)]">` `<lido:displayObject>` `<xsl:attribute name="lido:label">Collection title statement</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:displayObject>` `</xsl:for-each>` `</xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                  <lido:object>
                    <xsl:for-each select="onix:CollectionIdentifier/onix:IDValue">
                      <lido:objectID>
                        <xsl:attribute name="lido:type">
                          <xsl:for-each select="../onix:CollectionIdentifierType">
                            <xsl:if test="position() = 1">
                              <xsl:variable name="idx135" select="index-
of($map134/map, normalize-space())"/>
                              <xsl:choose>
                                <xsl:when test="$idx135 &gt; 0">
                                  <xsl:value-of
select="$map134/map[$idx135]/@value"/>
                                </xsl:when>
                                <xsl:otherwise>
                                  <xsl:value-of select="."/>
                                </xsl:otherwise>
                              </xsl:choose>
                            </xsl:if>
                          </xsl:for-each>
                        </xsl:attribute>
```

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="(../onix:IDTypeName)">` `<xsl:attribute name="lido:label">` `<xsl:for-each select="../onix:IDTypeName">` `<xsl:if test="position() = 1">` `<xsl:value-of select="."/>` `</xsl:if>` `</xsl:for-each>` `</xsl:attribute>` `</xsl:if>` `<xsl:value-of select="."/>` `</lido:objectID>` `</xsl:for-each>`	
		`<lido:objectNote>` `<xsl:attribute name="lido:type">Collection title</xsl:attribute>` `<xsl:for-each select="onix:TitleDetail/onix:TitleElement/onix:TitlePrefix">` `<xsl:value-of select="."/>` `</xsl:for-each> <xsl:for-each select="onix:TitleDetail/onix:TitleElement/onix:TitleWithoutPrefix">` `<xsl:value-of select="."/>` `</xsl:for-each>` `</lido:objectNote>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:ContributorStatement">`
		`  <lido:objectNote>`
		`    <xsl:attribute name="lido:type">Contributor statement</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectNote>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:CollectionType">`
		`  <xsl:variable name="idx137" select="index-of($map136/map, normalize-space())"/>`
		`  <xsl:choose>`
		`    <xsl:when test="$idx137 &gt; 0">`
		`      <lido:objectNote>`
		`        <xsl:value-of select="$map136/map[$idx137]/@value"/>`
		`      </lido:objectNote>`
		`    </xsl:when>`
		`    <xsl:otherwise>`
		`      <lido:objectNote>`
		`        <xsl:attribute name="lido:label">Collection type</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:objectNote>`
		`    </xsl:otherwise></xsl:choose></xsl:for-each>`
		`</lido:object></lido:relatedWork></lido:relatedWorkSet>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:DescriptiveDetail/onix:ProductPart">`
		`  <lido:relatedWorkSet>`
		`    <xsl:if test="(onix:PrimaryPart)">`
		`      <xsl:attribute name="lido:sortorder">1</xsl:attribute>`
		`    </xsl:if>`
		`    <lido:relatedWork>`
		`      <lido:object>`	

		`<xsl:for-each select="onix:ProductIdentifier/onix:IDValue">`
		`  <lido:objectID>`
		`    <xsl:attribute name="lido:type">`
		`      <xsl:for-each select="../onix:ProductIDType">`
		`        <xsl:if test="position() = 1">`
		`          <xsl:variable name="idx139" select="index-`
		`of($map138/map, normalize-space())"/>`
		`          <xsl:choose>`
		`            <xsl:when test="$idx139 &gt; 0">`
		`              <xsl:value-of`
		`select="$map138/map[$idx139]/@value"/>`
		`            </xsl:when>`
		`            <xsl:otherwise>`
		`              <xsl:value-of select="."/>`
		`            </xsl:otherwise></xsl:choose></xsl:if></xsl:for-`
		`each>`
		`    </xsl:attribute>`
		`    <xsl:attribute name="lido:label">Product`
		`identifier</xsl:attribute>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectID>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:ProductFormDetail"> <lido:objectNote> <xsl:value-of select="."/> </lido:objectNote> </xsl:for-each>`	
		`<xsl:if test="onix:ProductFormFeature/onix:ProductFormFeatureType = '01'"> <xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '01']"> <xsl:variable name="idx141" select="index-of($map140/map, normalize-space())"/> <xsl:choose> <xsl:when test="$idx141 &gt; 0"> <lido:objectNote> <xsl:value-of select="$map140/map[$idx141]/@value"/> </lido:objectNote> </xsl:when> <xsl:otherwise> <lido:objectNote> <xsl:value-of select="."/> </lido:objectNote> </xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="onix:ProductFormFeature/onix:ProductFormFeatureType = '02'">`
		`<xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '02']">`
		`<xsl:variable name="idx143" select="index-of($map142/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx143 &gt; 0">`
		`<lido:objectNote>`
		`<xsl:value-of select="$map142/map[$idx143]/@value"/>`
		`</lido:objectNote>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:objectNote>`
		`<xsl:value-of select="."/>`
		`</lido:objectNote>`
		`</xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="onix:ProductFormFeature/onix:ProductFormFeatureType = '04'">` `<xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '04']">` `<xsl:variable name="idx145" select="index-of($map144/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx145 &gt; 0">` `<lido:objectNote>` `<xsl:value-of select="$map144/map[$idx145]/@value"/>` `</lido:objectNote>` `</xsl:when>` `<xsl:otherwise>` `<lido:objectNote>` `<xsl:value-of select="."/>` `</lido:objectNote>` `</xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="onix:ProductFormFeature/onix:ProductFormFeatureType = '05'">`
		`<xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '05']">`
		`<xsl:variable name="idx147" select="index-of($map146/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx147 &gt; 0">`
		`<lido:objectNote>`
		`<xsl:value-of select="$map146/map[$idx147]/@value"/>`
		`</lido:objectNote>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:objectNote>`
		`<xsl:value-of select="."/>`
		`</lido:objectNote>`
		`</xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="onix:ProductFormFeature/onix:ProductFormFeatureType = '06'">` `<xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '06']">` `<xsl:variable name="idx149" select="index-of($map148/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx149 &gt; 0">` `<lido:objectNote>` `<xsl:value-of select="$map148/map[$idx149]/@value"/>` `</lido:objectNote>` `</xsl:when>` `<xsl:otherwise>` `<lido:objectNote>` `<xsl:value-of select="."/>` `</lido:objectNote>` `</xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		<xsl:if test="onix:ProductFormFeature/onix:ProductFormFeatureType = '09'"> <xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '09']"> <xsl:variable name="idx151" select="index-of($map150/map, normalize-space())"/> <xsl:choose> <xsl:when test="$idx151 &gt; 0"> <lido:objectNote> <xsl:value-of select="$map150/map[$idx151]/@value"/> </lido:objectNote> </xsl:when> <xsl:otherwise> <lido:objectNote> <xsl:value-of select="."/> </lido:objectNote> </xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="onix:ProductFormFeature/onix:ProductFormFeatureType = '12'">` `<xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '12']">` `<xsl:variable name="idx153" select="index-of($map152/map, normalize-space())"/>` `<xsl:choose>` `<xsl:when test="$idx153 &gt; 0">` `<lido:objectNote>` `<xsl:value-of select="$map152/map[$idx153]/@value"/>` `</lido:objectNote>` `</xsl:when>` `<xsl:otherwise>` `<lido:objectNote>` `<xsl:value-of select="."/>` `</lido:objectNote>` `</xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:if test="onix:ProductFormFeature/onix:ProductFormFeatureType = '13'"> <xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureValue[../onix:ProductFormFeatureType = '13']"> <xsl:variable name="idx155" select="index-of($map154/map, normalize-space())"/> <xsl:choose> <xsl:when test="$idx155 &gt; 0"> <lido:objectNote> <xsl:value-of select="$map154/map[$idx155]/@value"/> </lido:objectNote> </xsl:when> <xsl:otherwise> <lido:objectNote> <xsl:value-of select="."/> </lido:objectNote> </xsl:otherwise></xsl:choose></xsl:for-each></xsl:if>`	
		`<xsl:for-each select="onix:ProductFormFeature/onix:ProductFormFeatureDescription"> <lido:objectNote> <xsl:value-of select="."/> </lido:objectNote> </xsl:for-each>`	
		`<xsl:for-each select="onix:ProductFormDescription"> <lido:objectNote> <xsl:value-of select="."/> </lido:objectNote> </xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:NumberOfItemsOfThisForm">`
		`  <lido:objectNote>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectNote>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:NumberOfCopies">`
		`  <lido:objectNote>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectNote>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:CountryOfManufacture">`
		`  <lido:objectNote>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectNote>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:ProductContentType">`
		`  <lido:objectNote>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectNote>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:ProductForm">`
		`  <lido:objectNote>`
		`    <xsl:value-of select="."/>`
		`  </lido:objectNote>`
		`</xsl:for-each>`
		`</lido:object></lido:relatedWork></lido:relatedWorkSet>`
		`</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:RelatedMaterial/onix:RelatedProduct">`
		`<lido:relatedWorkSet>`
		`<lido:relatedWork>`
		`<lido:object>`	

```
            <xsl:for-each select="onix:RelatedMaterial/onix:RelatedProduct">
              <lido:relatedWorkSet>
                <lido:relatedWork>
                  <lido:object>
```

```
                    <xsl:for-each select="onix:ProductIdentifier/onix:IDValue">
                      <lido:objectID>
                        <xsl:attribute name="lido:type">
                          <xsl:for-each select="../onix:ProductIDType">
                            <xsl:if test="position() = 1">
                              <xsl:variable name="idx157" select="index-
of($map156/map, normalize-space())"/>
                              <xsl:choose>
                                <xsl:when test="$idx157 &gt; 0">
                                  <xsl:value-of
select="$map156/map[$idx157]/@value"/>
                                </xsl:when>
                                <xsl:otherwise>
                                  <xsl:value-of select="."/>
                                </xsl:otherwise></xsl:choose></xsl:if></xsl:for-
each>
                        </xsl:attribute>
                        <xsl:attribute name="lido:label">Product
identifier</xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:objectID>
                    </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:ProductFormDetail">`
		`<xsl:variable name="idx159" select="index-of($map158/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx159 &gt; 0">`
		`<lido:objectNote>`
		`<xsl:value-of select="$map158/map[$idx159]/@value"/>`
		`</lido:objectNote>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:objectNote>`
		`<xsl:attribute name="lido:label">Product form detail</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:objectNote>`
		`</xsl:otherwise></xsl:choose></xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:ProductForm">`
		`<xsl:variable name="idx161" select="index-of($map160/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx161 &gt; 0">`
		`<lido:objectNote>`
		`<xsl:value-of select="$map160/map[$idx161]/@value"/>`
		`</lido:objectNote>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:objectNote>`
		`<xsl:attribute name="lido:label">Product form</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:objectNote>`
		`</xsl:otherwise></xsl:choose></xsl:for-each>`
		`</lido:object></lido:relatedWork>`	

		`<lido:relatedWorkRelType>`
		`<xsl:for-each select="onix:ProductRelationCode">`
		`<lido:conceptID>`
		`<xsl:attribute name="lido:type">Local</xsl:attribute>`
		`<xsl:attribute name="lido:label">Product relation code</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:conceptID>`
		`</xsl:for-each>`
			Unfortunately there is no structural mapping to relatedWorkRelType in MINT so there is currently no direct correlation between conceptID and term elements taken from the same ONIX relation type (this should be unproblematic once SKOS links are added for IDs and terms).

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:ProductRelationCode">`
		`  <xsl:variable name="idx163" select="index-of($map162/map,`
		`normalize-space())"/>`
		`  <xsl:choose>`
		`    <xsl:when test="$idx163 &gt; 0">`
		`      <lido:term>`
		`        <xsl:value-of select="$map162/map[$idx163]/@value"/>`
		`      </lido:term>`
		`    </xsl:when>`
		`    <xsl:otherwise>`
		`      <lido:term>`
		`        <xsl:attribute`
		`name="lido:addedSearchTerm">no</xsl:attribute>`
		`        <xsl:attribute name="lido:label">Product`
		`relation</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:term>`
		`    </xsl:otherwise></xsl:choose></xsl:for-each>`
		`</lido:relatedWorkRelType></lido:relatedWorkSet>`
		`</xsl:for-each>`	

		`<xsl:for-each select="onix:RelatedMaterial/onix:RelatedWork">`
		`  <lido:relatedWorkSet>`
		`    <lido:relatedWork>`
		`      <lido:object>`	

LIDO section	LIDO subsection	XSLT	Comments

```
                    <xsl:for-each select="onix:WorkIdentifier/onix:IDValue">
                      <lido:objectID>
                        <xsl:attribute name="lido:type">
                          <xsl:for-each select="../onix:WorkIDType">
                            <xsl:if test="position() = 1">
                              <xsl:variable name="idx165" select="index-
of($map164/map, normalize-space())"/>
                              <xsl:choose>
                                <xsl:when test="$idx165 &gt; 0">
                                  <xsl:value-of
select="$map164/map[$idx165]/@value"/>
                                </xsl:when>
                                <xsl:otherwise>
                                  <xsl:value-of select="."/>
                                </xsl:otherwise></xsl:choose></xsl:if></xsl:for-
each>
                        </xsl:attribute>
                        <xsl:attribute name="lido:label">Work
identifier</xsl:attribute>
                        <xsl:value-of select="."/>
                      </lido:objectID>
                    </xsl:for-each></lido:object></lido:relatedWork>
```

LIDO section	LIDO subsection	XSLT	Comments
		`<lido:relatedWorkRelType>` `<xsl:for-each select="onix:WorkRelationCode">` `<lido:conceptID>` `<xsl:attribute name="lido:type">Local</xsl:attribute>` `<xsl:attribute name="lido:label">Work relation code</xsl:attribute>` `<xsl:value-of select="."/>` `</lido:conceptID>` `</xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:for-each select="onix:WorkRelationCode">`
		`<xsl:variable name="idx167" select="index-of($map166/map, normalize-space())"/>`
		`<xsl:choose>`
		`<xsl:when test="$idx167 &gt; 0">`
		`<lido:term>`
		`<xsl:value-of select="$map166/map[$idx167]/@value"/>`
		`</lido:term>`
		`</xsl:when>`
		`<xsl:otherwise>`
		`<lido:term>`
		`<xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`<xsl:attribute name="lido:label">Work relation</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:term>`
		`</xsl:otherwise></xsl:choose></xsl:for-each>`
		`</lido:relatedWorkRelType></lido:relatedWorkSet>`
		`</xsl:for-each>`
		
		`</lido:relatedWorksWrap></lido:objectRelationWrap></lido:descriptiveMetadata>`	

		`<lido:administrativeMetadata>`
		`<xsl:attribute name="xml:lang">en</xsl:attribute>`
		`<lido:rightsWorkWrap>`
		`<xsl:for-each select="onix:PublishingDetail/onix:CopyrightStatement">`
		`<lido:rightsWorkSet>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:rightsType>`
		`  <lido:term>`
		`    <xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>`
		`    <xsl:attribute name="xml:lang">en</xsl:attribute>Copyright</lido:term>`
		`</lido:rightsType>`	

		`<lido:rightsDate>`
		`  <xsl:for-each select="onix:CopyrightYear">`
		`    <xsl:if test="position() = 1">`
		`      <lido:earliestDate>`
		`        <xsl:attribute name="lido:type">`
		`          <xsl:for-each select="@dateformat">`
		`            <xsl:if test="position() = 1">`
		`              <xsl:value-of select="."/>`
		`            </xsl:if>`
		`          </xsl:for-each>`
		`        </xsl:attribute>`
		`        <xsl:attribute name="lido:label">Copyright date</xsl:attribute>`
		`        <xsl:value-of select="."/>`
		`      </lido:earliestDate>`
		`    </xsl:if>`
		`  </xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<xsl:for-each select="onix:CopyrightYear"> … </lido:rightsDate>`	
		`<lido:rightsHolder> … </xsl:attribute>`	

```
                <xsl:for-each select="onix:CopyrightYear">
                  <xsl:if test="position() = 1">
                    <lido:latestDate>
                      <xsl:attribute name="lido:type">
                        <xsl:for-each select="@dateformat">
                          <xsl:if test="position() = 1">
                            <xsl:value-of select="."/>
                          </xsl:if>
                        </xsl:for-each>
                      </xsl:attribute>
                      <xsl:attribute name="lido:label">Copyright
date</xsl:attribute>
                      <xsl:value-of select="."/>
                    </lido:latestDate>
                  </xsl:if>
                </xsl:for-each>
              </lido:rightsDate>
```

```
              <lido:rightsHolder>
                <xsl:for-each
select="onix:CopyrightOwner/onix:CopyrightOwnerIdentifier/onix:IDValue">
                  <lido:legalBodyID>
                    <xsl:attribute name="lido:type">
                      <xsl:for-each select="../onix:CopyrightOwnerIDType">
                        <xsl:if test="position() = 1">
                          <xsl:value-of select="."/>
                        </xsl:if>
                      </xsl:for-each>
                    </xsl:attribute>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<xsl:if test="(../onix:IDTypeName)">`
		`  <xsl:attribute name="lido:label">`
		`    <xsl:for-each select="../onix:IDTypeName">`
		`      <xsl:if test="position() = 1">`
		`        <xsl:value-of select="."/>`
		`      </xsl:if>`
		`    </xsl:for-each>`
		`  </xsl:attribute>`
		`</xsl:if>`
		`<xsl:value-of select="."/>`
		`</lido:legalBodyID>`
		`</xsl:for-each>`	

		`<lido:legalBodyName>`
		`  <xsl:for-each select="onix:CopyrightOwner/onix:CorporateName">`
		`    <lido:appellationValue>`
		`      <xsl:attribute name="lido:label">Corporate name</xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:appellationValue>`
		`  </xsl:for-each>`	

LIDO section	LIDO subsection	XSLT	Comments

```
          <xsl:for-each select="onix:CopyrightOwner/onix:PersonName">
            <lido:appellationValue>
              <xsl:attribute name="lido:label">Person name</xsl:attribute>
              <xsl:value-of select="."/>
            </lido:appellationValue>
          </xsl:for-each>
        </lido:legalBodyName>
      </lido:rightsHolder>
    </lido:rightsWorkSet>
  </xsl:for-each>
</lido:rightsWorkWrap>
```

```
<lido:recordWrap>
  <xsl:for-each select="onix:RecordReference">
    <lido:recordID>
      <xsl:attribute name="lido:type">Local</xsl:attribute>
      <xsl:value-of select="."/>
    </lido:recordID>
  </xsl:for-each>
```

```
  <lido:recordType>
    <xsl:for-each select="onix:DescriptiveDetail/onix:ProductComposition">
      <lido:conceptID>
        <xsl:attribute name="lido:type">Local</xsl:attribute>
        <xsl:value-of select="."/>
      </lido:conceptID>
    </xsl:for-each>
```

LIDO section	LIDO subsection	XSLT	Comments

```
            <xsl:for-each select="onix:DescriptiveDetail/onix:ProductComposition">
              <xsl:variable name="idx169" select="index-of($map168/map, normalize-
space())"/>
              <xsl:choose>
                <xsl:when test="$idx169 &gt; 0">
                  <lido:term>
                    <xsl:value-of select="$map168/map[$idx169]/@value"/>
                  </lido:term>
                </xsl:when>
                <xsl:otherwise>
                  <lido:term>
                    <xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>
                    <xsl:value-of select="."/>
                  </lido:term>
                </xsl:otherwise>
              </xsl:choose>
            </xsl:for-each>
          </lido:recordType>
```

LIDO section	LIDO subsection	XSLT	Comments

```
          <lido:recordSource>
            <xsl:attribute name="lido:type">europeana:dataProvider</xsl:attribute>
            <xsl:for-each select="onix:RecordSourceIdentifier/onix:IDValue">
              <lido:legalBodyID>
                <xsl:attribute name="lido:type">
                  <xsl:for-each select="../onix:RecordSourceIDType">
                    <xsl:if test="position() = 1">
                      <xsl:variable name="idx171" select="index-of($map170/map,
normalize-space())"/>
                      <xsl:choose>
                        <xsl:when test="$idx171 &gt; 0">
                          <xsl:value-of select="$map170/map[$idx171]/@value"/>
                        </xsl:when>
                        <xsl:otherwise>
                          <xsl:value-of select="."/>
                        </xsl:otherwise></xsl:choose></xsl:if></xsl:for-each>
                </xsl:attribute>
                <xsl:value-of select="."/>
              </lido:legalBodyID></xsl:for-each>
```

```
            <lido:legalBodyName>
              <xsl:for-each select="onix:RecordSourceName">
                <lido:appellationValue>
                  <xsl:value-of select="."/>
                </lido:appellationValue>
              </xsl:for-each>
            </lido:legalBodyName>
          </lido:recordSource>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:recordSource>`
		`  <xsl:for-each select="onix:RecordSourceIdentifier/onix:IDValue">`
		`    <lido:legalBodyID>`
		`      <xsl:attribute name="lido:type">`
		`        <xsl:for-each select="../onix:RecordSourceIDType">`
		`          <xsl:if test="position() = 1">`
		`            <xsl:variable name="idx173" select="index-of($map172/map, normalize-space())"/>`
		`            <xsl:choose>`
		`              <xsl:when test="$idx173 &gt; 0">`
		`                <xsl:value-of select="$map172/map[$idx173]/@value"/>`
		`              </xsl:when>`
		`              <xsl:otherwise>`
		`                <xsl:value-of select="."/>`
		`              </xsl:otherwise></xsl:choose></xsl:if></xsl:for-each>`
		`      </xsl:attribute>`
		`      <xsl:value-of select="."/>`
		`    </lido:legalBodyID></xsl:for-each>`	

		`  <lido:legalBodyName>`
		`    <xsl:for-each select="onix:RecordSourceName">`
		`      <lido:appellationValue>`
		`        <xsl:value-of select="."/>`
		`      </lido:appellationValue>`
		`    </xsl:for-each>`
		`  </lido:legalBodyName>`
		`</lido:recordSource>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<lido:recordRights> <lido:rightsType> <lido:conceptID> <xsl:attribute name="lido:type">URI</xsl:attribute>http://creativecommons.org/publicdomain/zero/1.0/</lido:conceptID> <lido:term> <xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>CC0 (mandatory only)</lido:term> </lido:rightsType> </lido:recordRights>`	
		`<lido:recordRights> <lido:rightsDate> <xsl:for-each select="../onix:Header/onix:SentDateTime"> <xsl:if test="position() = 1"> <lido:latestDate> <xsl:attribute name="lido:type">ISO 8601</xsl:attribute> <xsl:value-of select="."/> </lido:latestDate> </xsl:if> </xsl:for-each> </lido:rightsDate>`	

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:rightsHolder>`
		`<xsl:for-each select="onix:RecordSourceIdentifier/onix:IDValue">`
		`<lido:legalBodyID>`
		`<xsl:attribute name="lido:type">`
		`<xsl:for-each select="../onix:RecordSourceIDType">`
		`<xsl:if test="position() = 1">`
		`<xsl:value-of select="."/>`
		`</xsl:if>`
		`</xsl:for-each>`
		`</xsl:attribute>`
		`<xsl:attribute name="lido:label">Record source identifier</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:legalBodyID>`
		`</xsl:for-each>`	

		`<lido:legalBodyName>`
		`<xsl:for-each select="onix:RecordSourceName">`
		`<lido:appellationValue>`
		`<xsl:attribute name="lido:label">Record source name</xsl:attribute>`
		`<xsl:value-of select="."/>`
		`</lido:appellationValue>`
		`</xsl:for-each>`
		`</lido:legalBodyName>`
		`</lido:rightsHolder>`
		`</lido:recordRights>`	

LIDO section	LIDO subsection	XSLT	Comments

```
          <lido:recordRights>
            <lido:rightsDate>
              <xsl:for-each select="../onix:Header/onix:SentDateTime">
                <xsl:if test="position() = 1">
                  <lido:latestDate>
                    <xsl:attribute name="lido:type">ISO 8601</xsl:attribute>
                    <xsl:value-of select="."/>
                  </lido:latestDate>
                </xsl:if>
              </xsl:for-each>
            </lido:rightsDate>
```

```
            <lido:rightsHolder>
              <xsl:for-each
select="../onix:Header/onix:Sender/onix:SenderIdentifier/onix:IDValue">
                <lido:legalBodyID>
                  <xsl:attribute name="lido:type">
                    <xsl:for-each select="../onix:SenderIDType">
                      <xsl:if test="position() = 1">
                        <xsl:variable name="idx175" select="index-of($map174/map,
normalize-space())"/>
                        <xsl:choose>
                          <xsl:when test="$idx175 &gt; 0">
                            <xsl:value-of select="$map174/map[$idx175]/@value"/>
                          </xsl:when>
                          <xsl:otherwise>
                            <xsl:value-of select="."/>
                          </xsl:otherwise></xsl:choose></xsl:if></xsl:for-each>
                  </xsl:attribute>
```

LIDO section	LIDO subsection	XSLT	Comments

		`        <xsl:if test="(../onix:IDTypeName)">`
		`          <xsl:attribute name="lido:label">`
		`            <xsl:for-each select="../onix:IDTypeName">`
		`              <xsl:if test="position() = 1">`
		`                <xsl:value-of select="."/>`
		`              </xsl:if>`
		`            </xsl:for-each>`
		`          </xsl:attribute>`
		`        </xsl:if>`
		`        <xsl:value-of select="."/>`
		`      </lido:legalBodyID>`
		`    </xsl:for-each>`	

		`    <lido:legalBodyName>`
		`      <xsl:for-each select="../onix:Header/onix:Sender/onix:SenderName">`
		`        <lido:appellationValue>`
		`          <xsl:attribute name="lido:label">Sender name</xsl:attribute>`
		`          <xsl:value-of select="."/>`
		`        </lido:appellationValue>`
		`      </xsl:for-each>`
		`    </lido:legalBodyName>`
		`  </lido:rightsHolder>`
		`</lido:recordRights>`	

LIDO section	LIDO subsection	XSLT	Comments

		`    <xsl:for-each select="onix:ProductSupply/onix:Supplier/onix:Website">`
		`      <xsl:if test="(onix:WebsiteRole = '40')">`
		`        <lido:recordInfoSet>`
		`          <xsl:for-each select="onix:WebsiteLink">`
		`            <lido:recordInfoLink>`
		`              <xsl:value-of select="."/>`
		`            </lido:recordInfoLink>`
		`          </xsl:for-each>`
		`        </lido:recordInfoSet>`
		`      </xsl:if>`
		`    </xsl:for-each>`	

		`    <xsl:for-each select="onix:PublishingDetail/onix:Publisher/onix:Website">`
		`      <xsl:if test="onix:WebsiteRole = '40'">`
		`        <lido:recordInfoSet>`
		`          <xsl:for-each select="onix:WebsiteLink">`
		`            <lido:recordInfoLink>`
		`              <xsl:value-of select="."/>`
		`            </lido:recordInfoLink>`
		`          </xsl:for-each>`
		`        </lido:recordInfoSet>`
		`      </xsl:if>`
		`    </xsl:for-each>`
		`  </lido:recordWrap>`	

LIDO section	LIDO subsection	XSLT	Comments
		`<lido:resourceWrap>` `<lido:resourceSet>` `<lido:resourceRepresentation>` `<xsl:attribute name="lido:type">image_master</xsl:attribute>` `<xsl:if test="(onix:CollateralDetail/onix:SupportingResource/onix:ResourceContentType = '01') and (onix:CollateralDetail/onix:SupportingResource/onix:ResourceMode = '03') and (onix:CollateralDetail/onix:SupportingResource/onix:ResourceVersion/onix:ResourceForm = '01')">` `<xsl:for-each select="onix:CollateralDetail/onix:SupportingResource/onix:ResourceVersion/onix:ResourceLink[(../../onix:ResourceContentType = '01') and (../../onix:ResourceMode = '03') and (../onix:ResourceForm = '01')]">` `<xsl:if test="position() = 1">` `<lido:linkResource>` `<xsl:value-of select="."/>` `</lido:linkResource>` `</xsl:if></xsl:for-each></xsl:if>` `</lido:resourceRepresentation>`	

```
        <lido:resourceWrap>
          <lido:resourceSet>
            <lido:resourceRepresentation>
              <xsl:attribute name="lido:type">image_master</xsl:attribute>
              <xsl:if
test="(onix:CollateralDetail/onix:SupportingResource/onix:ResourceContentType =
'01') and (onix:CollateralDetail/onix:SupportingResource/onix:ResourceMode = '03')
and
(onix:CollateralDetail/onix:SupportingResource/onix:ResourceVersion/onix:ResourceFor
m = '01')">
                <xsl:for-each
select="onix:CollateralDetail/onix:SupportingResource/onix:ResourceVersion/onix:Reso
urceLink[(../../onix:ResourceContentType = '01') and (../../onix:ResourceMode =
'03') and (../onix:ResourceForm = '01')]">
                  <xsl:if test="position() = 1">
                    <lido:linkResource>
                      <xsl:value-of select="."/>
                    </lido:linkResource>
                  </xsl:if></xsl:for-each></xsl:if>
            </lido:resourceRepresentation>
```

LIDO section	LIDO subsection	XSLT	Comments

		`<lido:resourceRepresentation>`
		`  <xsl:attribute name="lido:type">image_thumb</xsl:attribute>`
		`  <xsl:if test="(onix:CollateralDetail/onix:SupportingResource/onix:ResourceVersion/onix:ResourceForm = '01') and (onix:CollateralDetail/onix:SupportingResource/onix:ResourceContentType = '01') and (onix:CollateralDetail/onix:SupportingResource/onix:ResourceMode = '03')">`
		`    <xsl:for-each select="onix:CollateralDetail/onix:SupportingResource/onix:ResourceVersion/onix:ResourceLink[(../onix:ResourceForm = '01') and (../../onix:ResourceContentType = '01') and (../../onix:ResourceMode = '03')]">`
		`      <xsl:if test="position() = 1">`
		`        <lido:linkResource>`
		`          <xsl:value-of select="."/>`
		`        </lido:linkResource>`
		`      </xsl:if></xsl:for-each></xsl:if>`
		`</lido:resourceRepresentation>`	

		`<lido:rightsResource>`
		`  <lido:rightsType>`
		`    <lido:term>`
		`      <xsl:attribute name="lido:addedSearchTerm">no</xsl:attribute>http://www.europeana.eu/rights/rr-p/</lido:term>`
		`  </lido:rightsType>`
		`</lido:rightsResource>`
		`</lido:resourceSet>`
		`</lido:resourceWrap>`
		`</lido:administrativeMetadata>`
		`</lido:lido>`
		`</xsl:template>`
		`</xsl:stylesheet>`	

19.4 ONIX ELEMENTS NOT MAPPED

ONIX Element, Composite or Group	Reason not included in mapping
<Barcode>	Technical detail mainly for retail sales use.
<TradeCategory>	Product classification for use within supply chain only (mainly for retailers).
<EpubTechnicalProtection> **<EpubUsageConstraint>**	Technical details mainly useful for (online?) retailers. In any case, usage conditions cannot be mapped to LIDO as yet.
<ProductClassification>	Tax code classification for products (mainly useful to distributors, wholesalers, retailers).
<ThesisType> **<ThesisPresentedTo>** **<ThesisYear>**	Most data contributions are not expected to be theses; in any case, these ONIX elements are not widely used.
Group P.8 Conference	Most data contributions are not expected to be conference proceedings.
<Complexity>	Complexity (of text for readers) is not commonly provided in ONIX records outside a small number of specific educational uses.
<ContentDetail>	The ContentDetail composite describes text items within the product; despite its potential interest for Linked Heritage, it is not widely used, nor would it map easily to LIDO.
<ProductContact>	The product contact's role is to support supply chain functions such as (B2B) sales and promotion, not for contact from end-customers. In any case, contact details cannot yet be expressed in LIDO.
<SupplyDetail> **- except: <Website>**	The SupplyDetail composite is used within the supply chain to enable B2B sales prior to retail. However, it can provide a <Website> composite containing a B2C retail link for the product which is mapped here. This composite may be necessary for a large-scale aggregation.

20 APPENDIX 5 – GUIDELINE FOR COMMERCIAL SECTOR DATA PROVIDERS IN LINKED HERITAGE

The following is an extract from the comprehensive response of the Linked Heritage project to Europeana's announced change of Data Exchange Agreement (DEA) to one including the condition that any data contributed to Europeana can be released under the CC0 rights waiver. The paper was approved by the project partners, and was implemented technically through a filter applied in the MINT system, at the stage of publication to Europeana. This section was developed by the Work Group 4 partners.

Any datasets used by Work Package 4 (WP4) will come from organisations outside of the Linked Heritage partners, as they will be sourced from the commercial sector.

20.1 CATEGORIES OF COMMERCIAL DATA CONTRIBUTIONS

The datasets will fall into 2 broad categories:

1. Test Data for use in creating mappings from commercial schemas to LIDO, ESE and EDM.

- Test Data will only be used within Linked Heritage, Work Group 4's controlled internal environment, seen only by Work Group 4 partners, and will not be published anywhere else, whether inside or outside the project. It will not be available on the Web. If required, the original Test Data files will be deleted from Work Group 4's systems once the mappings have been completed.
- Test Data will be used for development of appropriate metadata mapping schemas, in consultation with the data provider and relevant recognised industry specialists (who will not, however, receive the actual Test Data) and with reference to existing industry standards and best practice.
- Test Data should be as large a dataset as possible, and all data items ("records" or "messages") within it should be as full as possible – i.e. they should include as many of the possible data elements as possible, and ideally be "real" data identical to that used for business as usual. This will ensure that mappings are accurate and able to handle real data in future.

2. Prototype Data for contribution to Linked Heritage and publication to Europeana, under very strictly controlled conditions.

- Prototype Data will be used, as with the Test Data, within the controlled Work Group 4 environment, and additionally, a subset of each data item (i.e. only selected data elements from each data item, in agreement with the data provider) will be published to www.europeana.eu for demonstration and proof of concept. The Prototype will remain operative for a minimum of 30 days, after which, at a time specified by the data provider, Linked Heritage will instruct Europeana to remove the Prototype Data from their Web portal and all other data stores.
- The entire Prototype Data set (i.e. all elements of data items) will be used to test the mappings created with the Test Data as above. The Prototype Data subset will be used to
    - test the data publishing interface with Europeana;
    - and test the discovery of products within Europeana and the function of the link to buy the product in a retail environment
- Prototype Data can be a relatively small dataset (a small number of data items corresponding to a small selection of products). Each data item must, however, be a "real" item of product data as this prototype will Therefore the Prototype Data, for every product data item supplied, must include a link to either the provider's Website page for that specific product where it can be bought online, or an equivalent link to a retail/wholesale Website (selected by the data provider) for buying that specific product.
    - The operation of the Prototype will be publicised as agreed with the data provider, and the data provider is free to advertise the fact that their products will be discoverable within Europeana during the agreed Prototype operating period.

20.2 SIGNATURE OF THE DATA EXCHANGE AGREEMENT

Every data provider will need to sign the new DEA (http://version1.europeana.eu/web/europeana-project/newagreement/) directly with Europeana, if (and only if) they provide Prototype Data.

Therefore , each data provider should ensure that they understand the terms of this agreement and are able and willing to allow Europeana to publish the Prototype Data subset of elements on the open Web (through the www.europeana.eu/ search portal) and as Linked Open Data (http://pro.europeana.eu/linked-open-data). As with the data in the search portal, we will request that Europeana removes the Prototype Data subset from their Linked Open Data (LOD) once the Prototype operating period has ended.

There is a small chance (given the relative size of the datasets in question compared with the whole Europeana LOD) that this Linked Open Data may have been harvested, integrated into some other Linked Data stores, and maybe republished elsewhere in the meantime, before it can be removed from the Europeana LOD. Data providers should also be aware of this possibility and able to accept it for this subset of Prototype Data elements.

21 APPENDIX 6 – EXAMPLE OF PROPOSED ENHANCED LIDO EXPRESSIONS

In this example, an ONIX title including a year is mapped as a non-typed concatenation to LIDO, and as two, typed appellationValue part elements with sortorder to the proposed, enhanced LIDO. Note that the enhanced LIDO is simply a more flexible revision of the existing version; the new enhanced schema will still validate all of the existing LIDO data.

Schema	XML

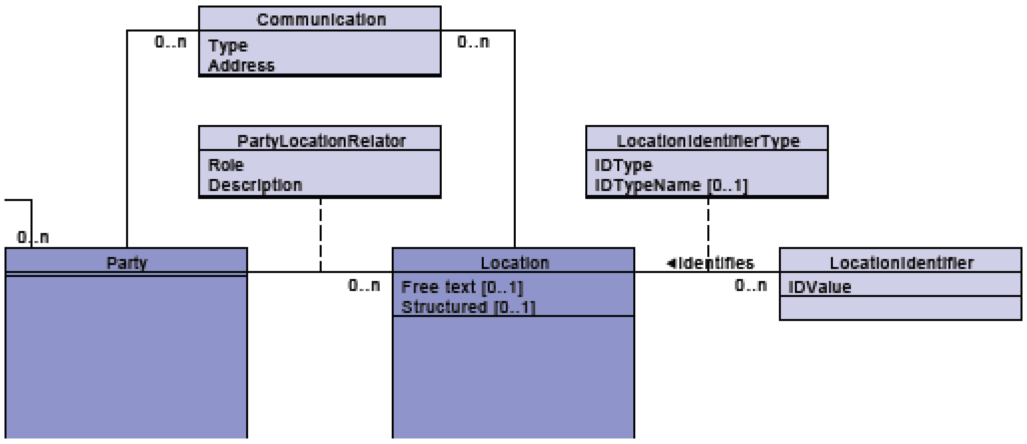
ONIX 3.0.1	`<onix:TitleElement>` `<onix:SequenceNumber>1</ onix:SequenceNumber >` `<onix:TitleElementLevel>01</onix:TitleElementLevel>` `<onix:TitleText>Annual Review of Heritage Publishing</onix:TitleText>` `</onix:TitleElement>`
	
	 `<onix:TitleElement>` `<onix:SequenceNumber>2</ onix:SequenceNumber >` `<onix:TitleElementLevel>01</onix:TitleElementLevel>` `<onix:YearOfAnnual>1963</onix:YearOfAnnual>` `</onix:TitleElement>`

LIDO	`[no mapping, or concatenation of some kind as in example below]`
	
	 `<lido:titleWrap>` `<lido:titleSet>` `<lido:appellationValue> Annual Review of Heritage Publishing (1963)</lido:appellationValue>` `</lido:titleSet>` `</lido:titleWrap>`

LIDO revised	`<lido:titleWrap>` `<lido:titleSet>` `<lido:appellationValue lido:label ="Title text " lido:sortorder=”1”>` `Annual Review of Heritage Publishing` `</lido:appellationValue>` `<lido:appellationValue lido:label ="Year of annual" lido:sortorder=”2”>` `1963` `</lido:appellationValue>` `</lido:titleSet>` `</lido:titleWrap>`

22 APPENDIX 7 – DRAFT GENERALISED CONTACT DETAILS MODEL

The diagram below is a small part of the draft generalised name and address model under development for the ONIX for Serials "toolkit" of messages and data structures[153]. This is a high-level model that might inform development of further communication details in extensions to LIDO.



Existing schemas containing granular contact details information:

ONIX for Publication Licences: http://www.editeur.org/21/ONIX-PL/

EDItX: http://www.editeur.org/52/Consumer-Direct-Fulfilment/ (order message)

PLUS: http://ns.useplus.org/ldf/LDFXML-1_2_0-Schema.xsd

[153] See overview of ONIX for Serials at http://www.editeur.org/17/ONIX-for-Serials/