LMER

Long-term preservation Metadata for Electronic Resources

Version: 1.2
Status: 7. April 2005
URN: urn:nbn:de:1111- 2005051906

Die Deutsche Bibliothek

Editor: Tobias Steinke

Translation into English: Dr. Thomas Wollschläger

Table of Contents

Introduction

I. Motivation

Electronic documents are a new class of media for libraries. Its contents, however, are generally comparable to ordinary books and publications. Thus, for content related descriptions the common bibliographic data that are being exchanged via data formats such as MAB2 or MARC21 can be used. Irrespective of that, electronic documents require further metadata that are especially crucial for the long-term preservation. While bibliographic data are stored in a catalogue system to support thematic enquiries of users, the technical metadata are to be stored directly with the electronic documents within a depot system. There they can be used for applying long-term preservation methods (emulation and migration).

The present document is to define these technical metadata, especially concerning long-term preservation. It attempts to meet the state of research on the subject of long-term preservation and, at the same time, to be relevant in practice and close to implementation.

It is generally undisputed that modern metadata should be described in XML, because this syntax format has established itself as a worldwide standard. For the detailed design, there are several theoretical preparatory works (to name paramountly the European project NEDLIB and the OCLC/RLG working group "Preservation Metadata Framework"). Concerning the practical elaboration, two approaches are mainly relevant: PREMIS[1], a very new approach (dating from the beginning of 2005) to define an international standard, and the "Preservation Metadata: Metadata Implementation Schema"[2] of the National Library of New Zealand (dating from July 2003).

Based on this, the following document presents the Long-term preservation Metadata for Electronic Resources (LMER) in which especially the data model from New Zealand is used as groundwork. The aim is an universal format for technical metadata, syntactically described by an XML schema. LMER is not a general data model for long-term preservation metadata, rather it was designed for concrete use as an exchange format. There is no default procedure for storing and administrating these metadata within a depot system. LMER can be used independently as an exchange format for technical metadata as well as a part of other XML structures.

[1] http://www.oclc.org/research/projects/pmwg/
[2] http://www.natlib.govt.nz/en/whatsnew/4initiatives.html#meta

II. Concepts

Core of technical metadata

Each file type (e.g., image, video, audio, text) requires special technical metadata. Which technical parameters accumulate in each case is subject to change depending on new developments in data processing. Therefore, it is a problem to develop an encyclopaedic metadata set that includes all possible metadata for all file types. Thus, LMER follows a different approach than the Metadata Implementation Schema of the National Library of New Zealand. It was intended to find a core of metadata that affects all file types. To include specific technical metadata, a field (xmlData) in the *lmerFile* part of LMER has been assigned to assimilate any XML metadata in another schema. This modular approach that has been made possible by the namespace concept of XML, makes LMER future-proof and more versatile. During the practical employment it is, however, necessary to find agreements on which metadata schema – if any – for a certain file type is to be applied.

References to describe the system environment

The description of the exact format and the necessary system environment is a key piece of information for long-term preservation, as it ensures the ability to reproduce the data in the originally intended form. This description includes many aspects of the hardware as well as the software. Because this information contains dependencies (e.g., a PDF document in a certain version can be displayed in all versions of the Acrobat Reader that are designed for it but the reader versions are available for certain operating systems that have certain requirements on the hardware), it is not useful to store it separately in files. Rather, an international database should be maintained that stores such dependencies, and references within the metadata of the different files should point to this database.

It therefore suffices to assign a unique identity from such a file format registry. If, for example, LMER references to the format "PDF 1.2", this would be a reference to such a central database that, on the other hand, when needed can return information about programs which can create and display that format, which operating systems are required by those programs and which hardware requirements are necessary. The other levels of the database can deliver reference capable notations, too. If, for example, LMER references to "OpenOffice.org 1.1" as the creation program, the database could name possible operating system and hardware requirements.

At present, there are several international attempts to create such databases of file formats. PRONOM[3] is already functioning but as yet without useful unique

[3] http://www.records.pro.gov.uk/pronom/

references. The national library of the Netherlands uses its own file format registry within the scope of its system DIAS, as part of the preservation manager.

Modularisation for the integration into other structures

Metadata for long-term preservation are normally used in the context of other metadata. To combine these different metadata into one object in one exchange format, further agreements are necessary. One of those is the Metadata Encoding and Transmission Standard (METS)[4]. METS describes the structure of an object and, additionally, can contain metadata parts of technical or textual origin. For integration into METS and similar XML metadata exchange formats, LMER is divided into several XML schemas. LMER can be used independently as an exchange format for technical metadata or as part of a more comprehensive exchange format.

As an example for an integration into METS, a possible procedure could be the following:

- Especially in the file section, METS contains certain attributes and fields that have equivalents in LMER. In favour of compatibility with other METS data, all LMER fields (including mandatory fields) that already have been described by compatible METS fields are being excluded.
- Data from the LMER part *lmerObject* are included in the METS section Administrative Metadata, in the sub-section Technical Metadata. That sub-section is referenced in the affected File Group description of the METS File Section.
- Data from the LMER part *lmerFile* are included in the METS section Administrative Metadata, in the sub-section Technical Metadata. That sub-section is referenced in the affected File description of the METS File Section.
- Data from the LMER part *lmerProcess* are included in the METS section Administrative Metadata, in the sub-section Digital Provenance. That sub-section is referenced in the affected File Group description (when referring to the object) or in the affected File description (when referring to a file) of the METS File Section.
- The LMER part *Modification* is excluded.
- References in LMER fields that are related to the LMER field fileIdentifier (startFile, linkedTo) now name instead the corresponding file description ID of the METS File Section.

[4] http://www.loc.gov/standards/mets/

III. Release Notes

Version 1.0: First official version.

Version 1.1: Internal version

- In accordance with the concept of modularisation, the XML schema is divided into several ones for the individual parts.
- Introduction of the fields linkedTo in the part *ImerFile* and status in the part *ImerObject*.

Version 1.2: Second official version

- The part *ImerProcess* can be a sub-part of *ImerObject* as well as of *ImerFile*.
- New fields in the part *ImerProcess*: oldMetadataRecordCreator, oldObjectIdentifier, oldObjectVersion.
- The fields format and transferFormat get the attribute FORMATREGISTRYNAME to specify the referenced file format registry.
- The fields fileChecksumType and transferChecksumType are replaced by adequate attributes to fileChecksum and transferChecksum.
- All fields that contain date/time statements are to be stated in the format ISO 8601 (XML data type xsd:dateTime).
- The following fields are repeatable now: format, transferFormat, fileChecksum, transferChecksum, groupIdentifier, formatInfos.

Reference

I. Overview of the elements

LMER is divided into the following parts that are geared to the Metadata Implementation Schema of New Zealand:

lmerObject: This includes metadata that corporately refer to all sub-files of the document. It also comprises the URN as a Persistent Identifier to establish a unique reference to the bibliographic metadata.

lmerProcess: Within these metadata, all technical changes to the object or to any file of the object are recorded. The part *lmerProcess* is either a sub-part of *lmerObject* or of *lmerFile*, and is repeatable in either case. For each change there is a new *lmerProcess* part.

lmerFile: For each file that belongs to the object, metadata describing its characteristics are stated here. These metadata are composed of general fields that are common to all file types, and of specific fields (e.g., the frame rate for videos) that are embedded into the field xmlData as specialised metadata from other Schemas. The part *lmerFile* is a sub-part of *lmerObject* and is repeatable. For each file belonging to an object, there is a new *lmerFile* part.

Metadata Modification: Within these metadata, all changes to the LMER metadata themselves are recorded. There are no changes accounted for that are not valid for the file itself.

II. Comprehensive reference description

Underlined fields are mandatory if LMER is not used as a part of another structure like METS. The part *lmerObject* is to be used invariably. If the numberOfFiles is larger than zero, it has to be the exact number of *lmerFile* parts. All other parts are optional. Mandatory fields within those parts only are needed if the part exists at all.

"NLNZ" within the line "Used standards" refers to corresponding fields in the Metadata Implementation Schema of the National Library of New Zealand. In brackets behind that, the corresponding field name and, if so, remarks concerning differences to the local definition are stated.

1. lmerObject

objectIdentifier

Definition: unequivocal identification of the object
Used standards: NLNZ (Object Identifier, but other data type)
Data type: string
Example: obj123
Commentary: Contrary to the Persistent Identifier (which must be globally unique), it is sufficient for this identification to be unique within the archiving institution's naming space. The field metadataRecordCreator provides a clue as to which context this ID derives from.

name

Definition: a human readable identification of the object
Used standards: NLNZ (Name of Object)
Data type: string
Example: Der fröhliche Jäger, Ausg. 06/1995
Commentary: Does not need to be unequivocal because not used for identification by machines.

persistentIdentifier

Definition: international unequivocal identification of the object
Used standards: NLNZ (Persistent Identifier)
Data type: string
Example: urn:nbn:de:gbv:104-opus-291
Commentary: Particularly, URN (see http://www.persistent-identifier.de/) are meant to be stated here but also other kinds of Persistent Identifiers (e.g., DOI) can find their place within this field. This identification can, for example, be used to define a connection to externally stored bibliographical metadata.

transferURL

Definition: URL to a transportable form of the object (combined within a file, e.g., packed as a ZIP file)
Used standards: URL
Data type: string
Example: ftp://ftp.bib-test.de/ddb/doc1234567890.zip
Commentary: That address does not need to be permanently valid.

transferFormat

Definition: identification of the file format to the field transferURL as an exact reference to an external database
Used standards: analogue to PREMIS (formatRegistryKey)
Data type: string
Example: PDF1.2
Commentary: This is not free-text but an as precise as possible format description that originates from a well-defined set of values. Which range of values is referenced, is stated by an attribute (REGISTRYNAME). That attribute can be a formal name or an URI. Whichever the significance of the field transferFormat is (e.g., if an URI), the attribute might be superfluous. The field is repeatable.

transferMimeType

Definition: MIME format of the file to the field transferURL
Used standards: MIME (see http://iana.org/assignements/media-types/index.html)
Data type: string
Example: application/zip
Commentary: -

transferChecksum

Definition: checksum of the file at transferURL
Used standards: CRC32, MD5, SHA-1, etc.
Data type: string
Example: 304ac95579c21f3498dfdff7117d3845220d34f
Commentary: An attribute (CHECKSUMTYPE) names the used checksum type. To identify the currently most used methods, exactly the following descriptions should be used within the attribute: CRC32, MD5, SHA-1. The field is repeatable.

groupIdentifier

Definition: an identification to mark objects belonging together
Used standards: NLNZ (Group Identifier, but other data type), PREMIS (preservationLevel)
Data type: string
Example: ddbszz11089
Commentary: ”Belonging together” mainly refers to technical aspects, e.g., pictures that were created under exactly the same conditions within a certain activity, or objects that are meant for cer-

tain preservation methods only (e.g., only emulation but no migration). The field is repeatable, i.e. each object can belong to several object groups.

objectVersion

Definition: If there are different archive objects of one original object (following an internal migration) then here is to be found a quick indicator for the version.
Used standards: -
Data type: string
Example: 2
Commentary: This means no textual version but only the alternate technical representation of the object itself. If the source file is being kept during a migration (that depends on the depot system), the new object (result of the migration) has its own data set of LMER data. Within this data set, the change description is recorded in the lmerProcess parts; but the LMER data of the still available but unchanged source file remains unmodified.

masterCreationDate

Definition: date/time when the archive object was created from the original object
Used standards: NLNZ (Preservation Master Creation Date)
Data type: date/time (ISO 8601)
Example: 2003-11-24T16:25:00
Commentary: As for a Web profile, this would be the date/time of the harvesting process.

metadataCreationDate

Definition: date/time of the creation of the technical metadata
Used standards: NLNZ (Date of Metadata Record Creation)
Data type: date/time (ISO 8601)
Example: 2003-11-24T16:28:00
Commentary: -

metadataRecordCreator

Definition: person or system or organisation by which the technical metadata were created
Used standards: NLNZ (Metadata Record Creator, but changed semantics)
Data type: string
Example: LMER Analyzer 1.2
Commentary: For a system, this should be specified exactly (including version number). This field names the context in within objectIdentifier is defined.

startFile

Definition: ID of a file that is the start file for several files

Used standards: similar to NLNZ (Target Indicator, but changed semantics and other data type)
Data type: string
Example: file0123
Commentary: The appropriate ID must be defined as fileIdentifier in a lmerFile part, no file name is to be found here.

numberOfFiles
Definition: overall number of files that belong to the object
Used standards: -
Data type: positive integer
Example: 5
Commentary: Must enumerate all lmerFile parts; folders do not count as files.

status
Definition: textual field to name an object related status
Used standards: -
Data type: string
Example: deleted
Commentary: This field acquires significance when using LMER as exchange format between different archiving systems.

comments
Definition: free text field for annotations
Used standards: NLNZ (Comments)
Data type: string
Example: object was made for testing purposes
Commentary: Only comments related to the entire object should be stated here.

2. lmerProcess

oldMetadataRecordCreator
Definition: person or system or organisation by which the technical metadata of the previous migration were created
Used standards: -
Data type: string
Example: LMER Analyzer 1.2
Commentary: For a system, this should be specified exactly (including version number). This field names the context in within oldObjectIdentifier is defined.

Old ObjectIdentifier
Definition: unequivocal identification of the object of the previous migration
Used standards: -

Data type: string
Example: obj122
Commentary: Other than the Persistent Identifier that is identical for each migration, the object ID changes for each conversion. The field oldMetadataRecordCreator provides a clue as to which context this ID derives from.

oldObjectVersion
Definition: version number of the previous migration
Used standards: -
Data type: string
Example: 1
Commentary: This version number helps to retrace the migration history.

purpose
Definition: describes the reason for the present modification
Used standards: NLNZ (Purpose)
Data type: string
Example: Format xyz is obsolete and is no longer supported.
Commentary: This is a free-text field that should be filled in as intelligibly as possible. Possibly, a list of defaults could be used.

processCreator
Definition: person or system which made the modifications
Used standards: NLNZ (Person / Agency Performing Process, but changed semantics)
Data type: string
Example: Preservation Toolbox V2.0
Commentary: -

permission
Definition: person, either a senior to the process creator or a supervisor to the system that was named as process creator
Used standards: NLNZ (Permission)
Data type: string
Example: Zimmermann, Robert
Commentary: This should always name a person, not yet another system.

permissionDate
Definition: date (and time) of a modification permission
Used standards: NLNZ
Data type: date/time (ISO 8601)
Example: 2003-11-24T17:30:00
Commentary: -

steps

Definition: description of the exact procedure of the modification
Used standards: NLNZ (Steps)
Data type: string
Example: 1. analysis of the format xyz, 2. loss-free migration into format zzz, 3. validation of the resulting format zzz
Commentary: When a standardised procedure has been applied, its description with a unequivocal reference (e.g., an URN) may be sufficient.

result
Definition: description of the status reached by the modification
Used standards: NLNZ (Result)
Data type: string
Example: All files with the format xyz, belonging to the object, were migrated loss-free into the format zzz.
Commentary: -

completionDate
Definition: date (and time) of the completion of the stated results
Used standards: NLNZ (Completion Date/Time)
Data type: date/time (ISO 8601)
Example: 2003-11-24T18:00:00
Commentary: -

comments
Definition: additional comments
Used standards: NLNZ (Comments)
Data type: string
Example: Because the migration has been completed loss-free, no copy of the original object is being kept.
Commentary: -

3. ImerFile

fileIdentifier
Definition: unequivocal identification of the appropriate file
Used standards: NLNZ (File Identifier)
Data type: string
Example: file0123
Commentary: This identification also acts as a link to the structural description (e.g., as in METS).

path
Definition: folder structure in which the file is to be put to be usable
Used standards: NLNZ (File Path)
Data type: string
Example: /docs/pdf/

Commentary: If this field is missing, the default is the path "/" (i.e., at the first level of the object). In case of several files that were not at the same level (the first level) of the object, this field is mandatory.

name

Definition: complete name of the file (without folder/s)
Used standards: NLNZ (Filename & Extension)
Data type: string
Example: dissertation.pdf
Commentary: -

size

Definition: file size in byte
Used standards: NLNZ (File Size)
Data type: positive integer
Example: 529123
Commentary: -

fileDateTime

Definition: date (and time) of the creation of the file
Used standards: NLNZ (File Date/Time)
Data type: date/Time (ISO 8601)
Example: 2002-12-24T12:00:00
Commentary: -

fileChecksum

Definition: checksum of the file
Used standards: CRC32, MD5, SHA-1, etc.
Data type: string
Example: 304ac95579c21f3498dfdff7117d3845220d34f
Commentary: An attribute (CHECKSUMTYPE) names the used checksum type. To identify the currently most used methods, exactly the following descriptions should be used within the attribute: CRC32, MD5, SHA-1. The field is repeatable.

mimeType

Definition: description of the file format according to the MIME standard
Used standards: NLNZ (MIME Type)
Data type: string
Example: application/pdf
Commentary: -

format

Definition: identification of the file format as an exact reference to an external database
Used standards: analogue PREMIS (formatRegistryKey)

Data type: string
Example: PDF1.2
Commentary: This is not free-text but an as precise as possible format description that originates from a well-defined set of values. Which range of values is referenced, is stated by an attribute (REGISTRYNAME). That attribute can be a formal name or an URI. Whichever the significance of the field transferFormat is (e.g., if an URI), the attribute might be superfluous. The field is repeatable.

formatInfos

Definition: additional information for an exact format identification
Used standards: -
Data type: string
Example: Linearized PDF, ISO PDF/X-1, ISO PDF/X-1a
Commentary: The value in the format field should normally be sufficient for a unequivocal identification. However, there could be important information on certain formats (e.g., because they were delivered by a tool like JHOVE) that can be stored in this field. The field is repeatable.

creatorApplication

Definition: identification of the original creation program of a file, ideally an exact reference to an external database
Used standards: PREMIS (creatingApplicationName, creatingApplicationVersion)
Data type: string
Example: OpenOffice.org 1.1
Commentary: Ideally, this should be not free-text but an as exact as possible program description that originates from a well-defined set of values. In any case, the program name should only be specified together with the version number.

viewerApplication

Definition: identification of an viewer application for the file, ideally an exact reference to an external database
Used standards: analog PREMIS (creatingApplicationName, creatingApplicationVersion)
Data type: string
Example: Adobe Reader 6.0
Commentary: Ideally, this should be not free-text but an as exact as possible program description that originates from a well-defined set of values. In any case, the program name should be used together with the version number only. This field is not exhaustive but should deliberately name a program that is known to correctly display the file.

linkedTo

Definition: file ID of another file within the same object that, when modified, causes a modification of the present file as well
Used standards: NLNZ (File Identifier)
Data type: string
Example: file0124
Commentary: During migrations, this field should help track necessary dependencies. E.g., a HTML file is dependent of all files linked to it, because a migration usually causes a change of the file name extensions, and therefore of the appropriate links as well. The dependencies only mean those to files of the same object, not to files of another object. The field is repeatable.

comments

Definition: additional comments
Used standards: -
Data type: string
Example: This file may also be displayed correctly with the Acrobat Reader 5.0 version.
Commentary: -

category

Definition: identification of the file type and additionally indicator for the following metadata
Used standards: DINI recommendation on how to denote the document type during the OAI set creation (see http://www.dini.de/dokumente.php)
Data type: string
Example: text
Commentary: The range of values is {text, notes, image, audio, video, multimedia, data, binary}

In the following, specific metadata for the stated category can be listed. As for images, for example MIX[5] can be used, as for texts, textMD[6]. Those metadata are not part of LMER and therefore belong to a different XML namespace.

xmlData

Definition: section with technical metadata, depending on the format
Used standards: XML metadata
Data type: XML data
Example: <mix> ... </mix>
Commentary: The mandatory attribute MDTYPE determines the exact name of the used metadata. There can be several sections with specific technical metadata, therefore the field is repeatable.

[5] http://www.loc.gov/standards/mix/
[6] http://dlib.nyu.edu/METS/textmd.htm

4. lmerModification

modifier

Definition: person or system which has applied the modifications

Used standards: NLNZ (Metadata Record Modifier)
Data type: string
Example: Preservation Toolbox V2.0
Commentary: -

dateTime

Definition: date (and time) of the modification
Used standards: NLNZ (Date/Time)
Data type: date/time (ISO 8601)
Example: 2003-11-24T18:00:00
Commentary: All modifications to the metadata (except of self-referencing), including the non LMER metadata have to be noted. The metadata must, however, exist within the same XML document.

fieldModified

Definition: exact description of the modified XML field
Used standards: NLNZ (Filed Modified, but other data type), XPath (see http://www.w3.org/TR/xpath)
Data type: string
Example: /mets/amdSec/lmer/object/process[@id=P1234]/steps
Commentary: The statement in XPath only works within an XML structure. When stored within a database, the metadata fields have to be referenced by internal identifiers.

data

Definition: value of the modified field before the modification
Used standards: NLNZ (Data Modified)
Data type: string
Example: migration
Commentary: The value within “string” might be to convert.

comments

Definition: additional comments
Used standards: -
Data type: string
Example: Description had been too vague before.
Commentary: A reason for the modification may be stated here as well.

Contact Person

Tobias Steinke, Die Deutsche Bibliothek, Deutsche Bibliothek Frankfurt am Main
E-Mail: steinke@dbf.ddb.de