SEMANTIC UNITS PERTAINING TO OBJECTS

Object entity

- Aggregates characteristics relevant to the preservation management of the object
- Semantic units may not all be applicable to each type of object (representation, file, bitstream)

Object entity

- Main types of information
  - 1.1. identifier
  - 1.5. object characteristics
  - 1.5.5. creation information
  - 1.8. software and hardware envmnt.
  - 1.9. digital signatures
  - 1.10. relationships to other objects
  - 1.11-13. links to other types of entity

1.2. objectCategory

- **objectCategory**
  - Values: representation, file, bitstream

1.3. preservationLevel

- **What preservation treatment/strategy the repository plans for this object**
  - Varying preservation options dependent on factors such as value, uniqueness, preservability of format
  - A business rule only relevant in a given repository
  - Optional for representation and file

1.3. preservationLevel

- **preservationLevelValue**
  - Examples: full, bit-level, fully supported with future migration

1.3. preservationLevel

- Additional (optional) semantic units:
  - Role: specifies context, e.g. if more than one
    - Examples: intention, requirement or capability
  - Rationale: important, when preservationLevelValue differs from usual repository policy, e.g. in case of a defective file.
  - Date: Date and Time when the preservationLevel was assigned to the object

1.4. significantProperties

- Applicable to representation, file and bitstream
- Characteristics subjectively considered important
  – e.g embedded JavaScript in PDF might be considered as important while Links in PDF are considered as unimportant and need not be preserved
- May help to measure preservation success
- Container for components:
  - significantPropertiesType - object property
  - significantPropertiesValue
  - significantPropertiesExtension

1.4. significantProperties

- May apply to all objects of a certain class or may be unique to each individual object
- Determine business rules of the repository
- Not an intrinsic property of an object;
  a particular archive's assessment of which of the object's properties need to persist over time;
  context specific
- Related to the preservation strategy chosen by the archive

1.4. significantProperties

- Listing significant properties implies that the repository plans to preserve those properties and would note any modifications to them in eventOutcome
- Further work is needed in determining and describing significant properties

Examples of significantProperties

Example 1:
significantPropertiesType = “behavior”
significantPropertiesValue = “editable”

Example 2:
significantPropertiesType = “page width”
significantPropertiesValue = “210 mm”

Example 3, a TIFF file:
significantPropertiesType = “Color space”
significantPropertiesValue = “Color accuracy (Adobe RGB 1998)”

Extension containers (general)

(e.g. significantPropertiesExtension, creatingApplicationExtension…)

- New in Premis 2.0
- Contains externally defined semantic units
- Allows to extend PREMIS with semantic units which are more granular, non-core or out of scope of the PREMIS data dictionary
- Data in the container may replace, refine or be additional to the appropriate PREMIS semantic unit
- One schema per extension; if more schemas are needed, the extension element needs to be repeated

1.5. objectCharacteristics

- Applicable only to file and bitstream (although some have needed it for representation)
- Technical properties common to all/most file formats,
  not format specific

1.5. objectCharacteristics

- Container for components:
  - compositionLevel
  - fixity
  - size
  - format
  - creatingApplication
  - Inhibitors

  - objectCharacteristicsExtension

1.5.2. fixity

- Information used to verify whether an object has been altered
- Compare message digests (“checksums”) calculated at different times
- Automatically calculated and recorded by repository

1.5.2. fixity

Container for

- **messageDigestAlgorithm***:*
  controlled vocabulary, examples:
  - SHA-1
  - MD5
- **messageDigest***:*
  output of message digest algorithm;
  checksum
- **messageDigestOriginator***:*
  agent that created original message digest;
  could be a string or a pointer

1.5.2. fixity

Example:

fixity

messageDigestAlgorithm= Adler-32

messageDigest= 7c9b35da

messageDigestOriginator= OCLC

1.5.4. format

- **Identifies the format of a file or bitstream**
- Container semantic unit
- Preservation activities depend on detailed and accurate knowledge about formats
- Should be ascertained by repository on ingest (for example, using JHOVE)
- May be a format name (**formatDesignation**) or a pointer into a registry (**formatRegistry**)
- Changed to repeatable in PREMIS version 2.0 to associate a format designation with a particular format registry

formatDesignation and formatRegistry

- **formatDesignation**
  - **Identifies the format of an object by formatname and formatversion**
  - Format may be a matter of opinion: Is it text, xml, or METS?
  - MIME type is widely used authority list
  - May need more granularity
  - may be multipart (tiff 6.0/geotiff)

formatDesignation and formatRegistry

- **formatRegistry**
  - **Identifies format by reference to an entry in a format registry**
  - Detailed specifications on formats may be contained in a future format registry
  - formatRegistry**Name**, formatRegistry**Key**, formatRegistry**Role**
  - Role includes purpose or expected use
- **formatNote** – free text

1.5.4. Examples of format

formatDesignation
formatName=eps
formatVersion=2.0

formatRegistry
formatRegistryName=PRONOM
formatRegistryKey=fmt/124
formatRegistryRole=Basic

1.5.4. Examples of format

formatDesignation
formatName=PDF
formatVersion=1.5

formatRegistry
formatRegistryName=
LC digital format descriptions
formatRegistryKey=fdd000123
formatRegistryRole=assessment

1.5.5. creatingApplication

- **Information about the application(s) which created a file/bitstream**
- Software bugs are not uncommon.
- May affect the integrity of content or create artifacts.
- In a repository it might be useful to search for all files created by a certain version of an application to fix them.

1.5.5. creatingApplication

- **creatingApplicationName**
- **creatingApplicationVersion**
- **dateCreatedByApplication**
  - Actual or approximated date and time when the object was created
- **creatingApplicationExtension**
  - Specified metadata schema can be included instead or in addition to PREMIS defined semantic units
  - Additional schema might contain values from a controlled list, point to a registry….

1.5.6. inhibitors

- **Features of the object intended to inhibit access, use or migration**
- It is necessary to record the kind of encryption and the access key to allow future use of the object
- Applicable to file and bitstream

1.5.6. inhibitors

- **inhibitorType**
  - Inhibitor method employed, e.g. “DES”, “password protection”
- **inhibitorTarget**
  - The content or function protected, e.g. “function: print”
- **inhibitorKey**
  - The decryption key or password
  - Needs to meet security requirements

1.5.6. inhibitors

Example:

inhibitors

inhibitorType=DES

inhibitorTarget=all content

inhibitorKey=[DES encryption key]

1.5.7. objectCharacteristicsExtension

- **Container to include externally defined semantic units – e.g. for more granularity.**
- Might contain format specific metadata for a file – e.g. technical metadata for still images (MIX)
- Not a replacement for units specified in PREMIS

1.5.1. compositionLevel

- **An indication of whether the object is subject to one or more processes of decoding or unbundling**
- How to describe layers of encodings so they can be correctly reversed
  - Treat each layer as a "composition level"
  - Repeat description of object characteristics for each composition level

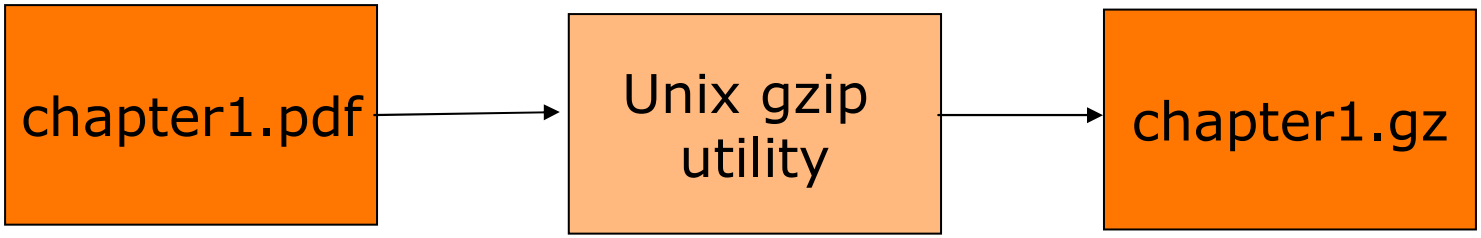
1.5.6. compositionLevel

- A file with no compression and no encryption has compositionLevel 0 (zero)
- Each layer of encoding results in new format and incremented compositionLevel
- **Only applies if object is encrypted or compressed**
- Value is an integer

Files again

- **FILE** = a named and ordered sequence of bytes that is known by an operating system.
  - chapter1.pdf
  - photo.tiff
  - mapofBerlin.jp2
- Can be zero or more bytes
- Has a file format
- Has access permissions and file system statistics such as size and modification date

But some files aren't that simple



- format = PDF
- size = 500,000 bytes
- messageDigest = [something]

- format = gzip
- size = 324,876 bytes
- messageDigest = [something else]

1.5.1. compositionLevel

chapter1.pdf			
compositionLevel			0
fixity	messageDigestAlgorithm		SHA-1
fixity	messageDigest		[big string]
fixity	messageDigestOriginato		Submitter
size			500000
format	formatDesigna-tion	formatName	PDF
format	formatDesigna-tion	formatVersion	1.2

chapter1.pdf.gz			
compositionLevel			1
fixity	messageDigestAlgorithm		SHA-1
fixity	messageDigest		[another string]
fixity	messageDigestOriginator		Repository
size			324876
format	formatDesigna-tion	formatName	gzip
format	formatDesigna-tion	formatVersion	1.2.3

1.5.1. compositionLevel

- Remember: Composition level increments only when you have a single file object with multiple successive encodings.

Creation information

1.5.5. creatingApplication

- **Container for information about the application and the context in which an object was created**
  - creatingApplicationName
  - creatingApplicationVersion
  - dateCreatedByApplication
  - creatingApplicationExtension
- Part of objectCharacteristics

Creation information

1.6. originalName

- **Name of object as submitted to or harvested by repository**
- Supplements repository supplied names
- Useful for identification of objects for clients or outside partners
- Applicable to files and representations

1.7. storage

- **How and where the object is stored**
- Container for
  contentLocation
  storageMedium
- May be repeated if more than one identical copies in different locations
  that are managed together

1.7. storage

- **contentLocation**
  - Information needed to retrieve a file from a system or a bitstream from within a file
  - Subunits type and value
  - Could be fully qualified path or identifier used by storage system; for bitstream a byte offset
- **storageMedium**
  - Physical medium on which the object is stored
  - Useful for media management (e.g. media migration)
  - May be name of system that knows the medium
  - Examples: hard disk, TSM

1.7. Example of storage

storage

contentLocation

contentLocationType=FDA

contentLocationValue=fda/prod/data/out/classa/DF-2005-001002

storageMedium=3590 [a type of tape unit]

1.8. Environment

- **What is needed to render or use an object**
  - Operating system
  - Application software
  - Computing resources
- Why is obligation optional?
  - Need for this information may differ in preservation strategies (e.g., may be unneeded for bit-level preservation)
  - We currently lack practical methods to collect and store this information

1.8. Environment

- Relevance to long-term preservation:
  Ability to render an object and interact with its content may depend on knowing these technical details

- Applies to all types of object (representation, file, bitstream)

1.8. Environment semantic units

- **environmentCharacteristic**
  - Multiple environments can support an object, but often not equally well
  - Suggested values: unspecified, known to work, minimum, recommended
  - Repository does not need to record all possible environments

1.8. Environment semantic units

- **environmentPurpose**
  - Use supported by the specified environment
  - Suggested values: render, edit
  - example: for x.pdf
    Adobe Acrobat (edit)
    Adobe Reader (render)

1.8. Environment semantic units (cont.)

- **software and hardware**
  - identify by name, version, type (broad category)
  - Many may apply; at least one should be recorded

- **dependency**
  - non-software component or file needed
  - dependency vs. swDependency
  - e.g. fonts, schemas, stylesheets
  - name and identifier

1.8. Environment semantic units (cont.)

- **environmentNote**
  - Any additional information
  - Should not be used as substitute for more rigorous description

- **environmentExtension**
  - Replace or extend PREMIS semantic units
  - In an operation environment a link to an appropriate system/emulator can be stored.

1.8. Environment example: ETD (PDF file)

- environmentCharacteristic=known to work
- environmentPurpose=render
- software/swName= Mozilla Firefox
- software/swVersion= 1.5
- software/swType=renderer
- swOtherInformation=requires swDependencies as plug-ins
- software/swDependency= Adobe Acrobat Reader 7.0
- software/swDependency= RealPlayer 10

1.8. Environment example: ETD (PDF file)

- software/swName= Windows NT
- software/swVersion=5.0 (2000)
- software/swType=operatingSystem

- hardware/hwName=Intel Pentium III
- hardware/hwType=processor

- dependency/dependencyName=Mathematica 5.2 True Type math fonts

1.8. Environment registries

- Information may be complex and increasingly granular
- Information often applies to whole class of objects
- PREMIS does not assume the existence of an environment registry, but defines the information that would be needed in one

1.8. Environment registries

- PRONOM has some elements of environment registry
  - for any file extension, gives list of software that can
    - create
    - render
    - identify
    - validate
    - extract metadata from

1.9. Digital signatures

- In a transaction,
  verifies the identity of the sender and
  that the file was unchanged in transmission.

- Some archives sign stored objects for verification of authenticity in the future.

1.9. Digital signatures

- PREMIS digital signature semantic units are based on W3C's *XML Signature Syntax and Processing*
  - *de facto* standard for encoding signature information
  - PREMIS adopts structure/semantics where possible
  - Some departures: e.g., PREMIS permits a given signature to be a property of only 1 object.

1.9. signatureInformation Container

- *Who signed it?*
  - signer (name or pointer to an Agent)

- *How was it signed?*
  - signatureInformationEncoding (e.g., Base64)
  - signatureMethod (e.g., DSA-SHA1)

1.9. signatureInformation Container

- *How can we validate it?*
  - signatureValidationRules (could be a pointer to documentation for the validation procedure)
  - signatureProperties (additional information)
  - keyInformation: the signer's public key and other info
    - Type: e.g., DSA, RSA, PGP, etc.
    - Other info: e.g., certificate, revocation list, etc.

- *And of course, the signature itself*

1.9. signatureInformation example

signatureInformation
signatureInformationEncoding=base64
signer=Florida Digital Archive
signatureMethod=RSA-SHA1
signatureValue=MC0CFFrVLtRlkMc3Daon4BqqnkhCOTFEALE=
signatureValidationRules=T1=C1
signatureProperties=2003-03-19T12:25:14-05:00
keyInformation
keyType=x509v3-sign-rsa2
keyValue=<DSAKeyValue>
keyvalue
</DSAKeyValue>