

# The Typed Object Model

Support for diverse formats

John Mark Ockerbloom

File Formats for Preservation Seminar

May 10, 2004

# Standard preservation formats are great, but...

- You may need to deal with other formats as well (at least for ingestion)
- You may be using particular profiles or abstractions on top of standard formats that also need support
- Even for standard formats, you may need tools for analyzing, converting into/out of format that aren't readily available locally
- TOM can help you with these problems...

# What is TOM?

- “Typed Object Model” (PhD thesis at CMU)
- A **model** for **identifying and describing data formats** (and information retrieval services) in well-defined, machine-processable manners
- A distributed **system** of “type brokers” that **maintain and interpret these descriptions**, and operate on data in the formats they describe
- Works with existing data formats, systems
  - No prerequisites for formats other than being byte sequences
  - Supports format migration (with controlled information loss), and functional emulation
  - Can deal with formats for end users or programs-- even if the program has never seen the format before
  - Has working implementations
    - » including Web-based file conversion service
- Current development funded by Andrew W. Mellon Foundation

# How can formats be described?

- **Syntactically:** What the bytes look like
  - E.g. BSDL, grammars
  - Verification: Signatures and other “magic number” checking
- **Structurally:** What data structures are present
  - E.g. DIDL, ASN.1 declarations, many specification documents
  - May assume particular underlying syntax (e.g XML Schemas), or abstract away from it (e.g. Dublin Core and OAIIS definitions)
- **Functionally:** What can be done with the format
  - E.g.APIs, crosswalks, PRONOM software registry
- **Practically:** Effective use and sustainability of format
  - E.g. support, quality, IP restrictions...
  
- **TOM focuses primarily on functional aspects**
  - Functions (extraction, resolution, conversion) can be defined, invoked
  - TOM also deals with syntax and structure to some extent
  - Other aspects may be dealt with through a general format registry

# What is a format?

- In TOM, it's a **type** combined with a sequence of **encodings** that represents the type
- A type describes the information contained in an **object** (a unit of data)
  - What is contained in the object (attributes)
    - » (e.g. a URL has a protocol part, a host/port part...)
  - How the object “behaves” when interpreted (operations)
    - » (e.g. a URL can be resolved to yield another object)
  - What constraints exist on the object (semantics)
    - » (e.g. a URL's port number must be non-negative)
- An encoding is a syntax for that information
  - it maps objects from one type to those of another type that represents it
- Fully-specified formats map down to bytes

# Format example: TAR

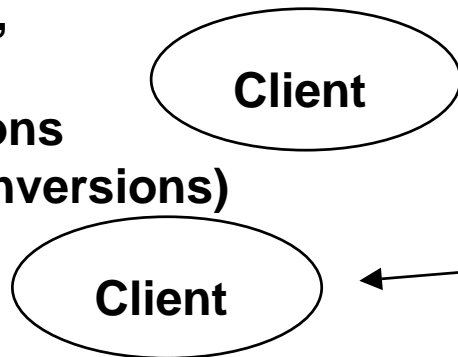
- A “TAR” type
  - has **attributes** like “pathnames of files in the archive”
  - has **operations** like “get the file with pathname n”
- ...is a subtype of an abstract “package” type
  - with attributes and operations like “get item names; get item with name n”, but without features specific to TAR
- ...can be encoded as a “byte sequence” type via the “POSIX tar syntax” method
  - Type+encodings = format. Format then has **conversions**
- Never seen TAR before? You can still use it...
  - ...by having a type broker identify and describe the format
  - ...by calling interface of the known “package” supertype
  - ...or by converting it to a format you know (like zip)
    - » with option to preserve “package” aspects of format
    - » or the aspects of any other supertype of TAR

# Multilayered format example: Dublin Core

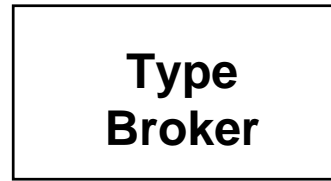
- A “Dublin Core” type
  - has attributes like “creators, titles...”
- ...encoded as an “XML” type via the “OAI-DC” method
  - has operations like “get elements with this Xpath...”
- ...encoded as a “Unicode code points” type via the “standard XML 1.0 syntax” method
  - has operations like “get the nth character code...”
- ...encoded as “byte sequence” via the “Unicode UTF-8” method
  - has attributes like “number of bytes...”
- The object can be interpreted at any of these abstraction levels
  - Just annotate the basic bytes with type, encoding names

# TOM as a distributed system

Clients get info on formats, request operations (e.g. conversions)

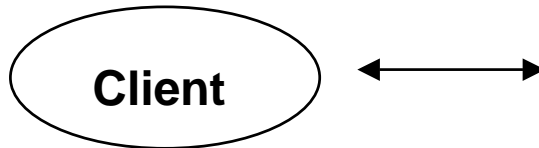


Brokers maintain info on formats, invoke servers for operations

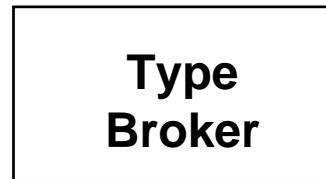


Servers implement operations

Clients can also register new formats, functions, server information...



Brokers can trade info, consult other brokers



# How are type, format, function definitions maintained?

- **Any type broker can originate a definition**
  - and others can copy it, peer-to-peer style
  - “Definitive” type brokers can also arise
- **Namespaces, owners mitigate conflicts**
  - Each broker owns DNS-based namespace for type names
  - Standard registries (“mime”, “dlf”) can get namespace
    - » Can adopt existing types via aliasing or copying
  - Each type has “owner” responsible for resolving naming conflicts, inconsistencies within the type
  - Owner preapproval not required for most registrations
- **Reliability maintained as standards evolve**
  - “Published” status locks essential type aspects
  - New types/formats can be defined with conversions from obsolete ones that they supplant

# Some limitations of TOM

- **Oriented to programmers (not librarians)**
  - Type+encodings notation may be confusing to laypeople
  - Defining good types requires familiarity with object-oriented design, careful specification of semantics
  - Limited tolerance for imprecision, exceptions
    - » sometimes that's a benefit at large scale, though
- **Not sufficient to cover all aspects of formats**
  - Documentation, tutorials, tools not part of TOM proper
  - Some kinds of information less not based on supertypes
  - Lacks expressive range of pure XML-based systems
- **Doesn't run itself**
  - For preservation, someone has to commit to **maintaining** comprehensive corpus of format information, services

# TOM and format registries

- **Global Digital Format Registry system initiative**
  - Would hold human-readable specifications, informal notes, and other documentation of various sorts
  - A testbed prototype is now available online
- **Current prototype interoperates with TOM**
  - Format registry records can refer to TOM specifications
  - In future, registry might also refer directly to TOM implementations of conversions or other services
  - Conversely, TOM type descriptions might refer to registry for more information, or for implementations of functions

# For more information

- **Web site:** <http://tom.library.upenn.edu/>
  - Documentation on TOM
  - Conversion service, type browser
  - Format registry demonstration (“Fred”)
  - Open source software for running your own TOM brokers/servers, defining new formats in TOM (coming soon!)
- **We welcome questions, suggestions, advice, collaboration...**
- **Email:** [ockerblo@pobox.upenn.edu](mailto:ockerblo@pobox.upenn.edu)