ARTICLES

D-Lib Magazine
February 2004

Volume 10 Number 2

ISSN 1082-9873

Using MPEG-21 DIP and NISO OpenURL for the Dynamic Dissemination of Complex Digital Objects in the Los Alamos National Laboratory Digital Library

Jeroen Bekaert
Los Alamos National Laboratory, Research Library and Ghent University, Faculty of Engineering
<jbekaert@lanl.gov>

Lyudmila Balakireva
Los Alamos National Laboratory, Research Library
<ludab@lanl.gov>

Patrick Hochstenbach
Los Alamos National Laboratory, Research Library
<hochsten@lanl.gov>

Herbert Van de Sompel
Los Alamos National Laboratory, Research Library
<herbertv@lanl.gov

Abstract

This paper focuses on the use of NISO OpenURL and MPEG-21 Digital Item Processing (DIP) to disseminate complex objects and their contained assets, in a repository architecture designed for the Research Library of the Los Alamos National Laboratory. In the architecture, the MPEG-21 Digital Item Declaration Language (DIDL) is used as the XML-based format to represent complex digital objects. Through an ingestion process, these objects are stored in a multitude of autonomous OAI-PMH repositories. An OAI-PMH compliant Repository Index keeps track of the creation and location of all those repositories, whereas an Identifier Resolver keeps track of the location of individual complex objects and contained assets. An MPEG-21 DIP Engine and an OpenURL Resolver facilitate the delivery of various disseminations of the stored objects. While these aspects of the architecture are described in the context of the LANL library, the paper will also briefly touch on their more general applicability.

1. Introduction

When compared to most academic and research libraries, the Research Library of the Los Alamos National Laboratory (LANL) follows a rather unique strategy with respect to providing access to digital scholarly information. The general trend in digital library services is to have users access externally hosted materials through third party services, federated through a locally hosted Web Portal. In order to be self-supporting with respect to mission-critical scholarly information, the LANL library acquires or licenses a vast collection of digital scholarly assets, hosts those assets locally, and makes them accessible through locally developed user services. The locally hosted assets include secondary data feeds from ISI, BIOSIS, Inspec, and primary information feeds from major scholarly publishers such as

Elsevier, Wiley, IOP, APS, etc.

At the time of writing, the collection of locally hosted assets amounts to approximately 5 Terabytes of raw materials. In addition to that, the LANL library is actively investigating the deployment of Institutional Repository capabilities to host locally created materials such as technical reports, datasets, videotaped presentations, etc. Also, research is underway to augment the locally hosted collection with materials gathered by focused Web crawling and to include logs detailing the usage of repository assets in the repository as assets in their own right [1,2]. Hosting, archiving and making accessible such a vast and heterogeneous collection of scholarly assets in a consistent and sustainable manner is a challenge that touches on many areas of Digital Library practice and research, including the identification of assets, the expression of relationships between assets, the representation of assets by means of complex object models, and methods to ingest and access stored assets.

Over the last year, the Digital Library Research and Prototyping Team of the LANL Research Library has worked on the design of a LANL Repository architecture aimed at ingesting, storing, and making accessible to downstream applications an ever growing heterogeneous digital collection. Also, a working prototype of the design has been implemented. While no claims are being made that the LANL Repository design or implementation are of a nature that merits a comparison with—say—the deliverables of the DSpace [3] or Fedora [4,5] projects, the authors do feel that the architecture has the following interesting properties that should be attractive for repository-related projects beyond the realm of the LANL Research Library:

1. The use of the MPEG-21 Digital Item Declaration Language (DIDL) to represent complex objects, as described in our previous D-Lib paper [6]. A reading of that paper is recommended to obtain a good understanding of this discussion.
2. The natively distributed nature of the architecture.
3. The use of a special technique—the XMLtape—to store and make accessible static collections of complex objects.
4. The multi-faceted use of the OAI-PMH to access stored content in incremental batches.
5. The use of NISO OpenURL to access stored content or various disseminations thereof.
6. The dynamic binding of dissemination methods to stored content.
7. The use of MPEG-21 Digital Item Processing (DIP) to execute different services associated with DIDs upon request of an agent.

Figure 1 introduces the major components of the LANL Repository architecture. As can be seen, the LANL Repository hosts a multitude of autonomous OAI-PMH [7] repositories, each of which stores complex digital objects represented using the MPEG-21 DIDL [8] format. The Repository Index keeps track of the creation of such autonomous OAI-PMH repositories as well as of their location. The Repository Index is exposed as an OAI-PMH repository in its own right. The OAI-PMH Federator exposes the whole LANL Repository as a single OAI-PMH repository. It interacts with other components of the architecture mainly using the OAI-PMH. The OAI-PMH Federator hides the complexity of the environment to downstream harvesters, and becomes their single point of access to harvest from the LANL Repository. The aforementioned components of the LANL Repository architecture are only briefly discussed in Section 3. They will be described in detail in a forthcoming paper that focuses on aspects (2), (3) and (4) above. This paper elaborates on properties (2), (5), (6), and (7) of the LANL Repository architecture by describing its use of MPEG-21 DIP [9] and NISO OpenURL [10] for the dynamic dissemination of the stored complex objects and their contained assets. To that end, Section 4 describes the Identifier Resolver, which contains identifiers of stored content, as well as the location of the OAI-PMH repository in which the content resides. The Identifier Resolver is populated through OAI-PMH harvesting and can be queried in a variety of ways, including the Handle protocol [11]. The DIP Engine, which works according to MPEG-21 principles, is introduced in the environment to facilitate the delivery of various disseminations of stored content. A high-level insight in the operation of the DIP Engine is provided in Section 5. Section 5 also describes the OpenURL Resolver, which provides a front-end to the LANL Repository from which various disseminations of individual assets contained in the LANL Repository can be obtained using requests compliant with the forthcoming NISO OpenURL Standard.

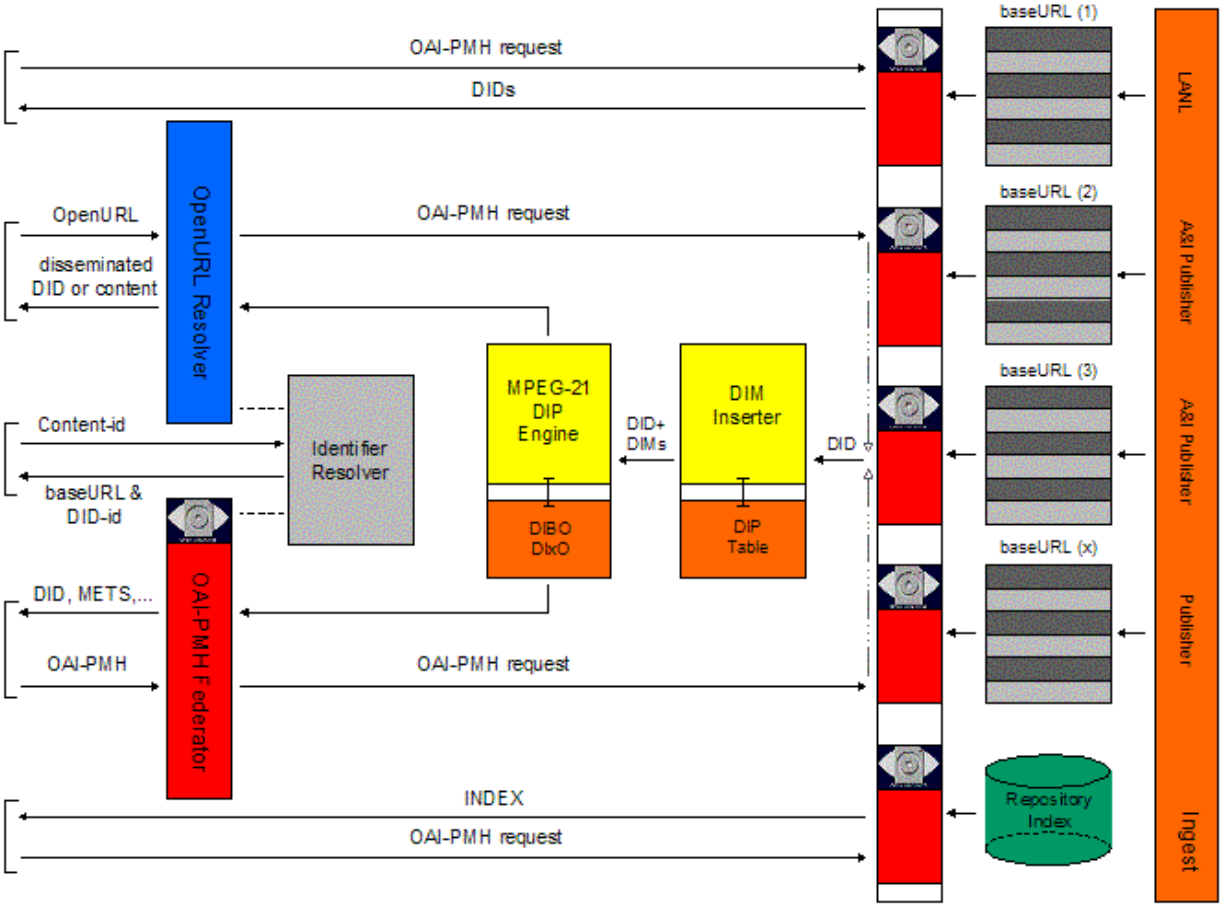

Figure 1: LANL Repository Architecture

2. Ingestion into the LANL repository

In many cases the delivered assets to be hosted in the LANL Repository are 'complex' in the sense that they consist of multiple individual datastreams that form a single logical unit. For example, a scholarly article may be delivered as a bundle that consists of metadata describing the article, the article itself in PDF and ASCII format, and the references made in the article expressed in XML. The complex nature of the assets led to an investigation regarding existing approaches to represent complex digital objects using XML wrappers, which resulted in the selection of the MPEG-21 Digital Item Declaration Language (DIDL) [8] as the sole way to store digital assets in the LANL Repository. DIDL introduces a set of abstract concepts that, together, form a well-defined data model for complex digital objects [note 1]. Based on those abstract concepts, DIDL defines a W3C XML Schema that provides broad flexibility and extensibility for the actual representation of compliant complex digital objects [note 2]. The DIDL data model recognizes the following entities, each of which has a corresponding XML element in the DIDL XML Schema:

- A ***Container*** is a grouping of ***Containers*** and/or ***Items***.
- An ***Item*** is a grouping of ***Items*** and/or ***Components***.
- A ***Component*** is a grouping of ***Resources***. Multiple ***Resources*** in the same ***Component*** are considered equivalent and consequently an agent may use any one of them.
- A ***Resource*** is an individual datastream.
- Secondary information pertaining to a ***Container***, an ***Item***, or a ***Component*** can be conveyed by means of a ***Descriptor***.

Digital assets to be hosted by the LANL Repository can, in principle, be obtained in a variety of ways including FTP, OAI-PMH harvesting, Web crawling and delivery on physical media. A prototype ingestion process has been developed that turns each obtained asset into an autonomous XML document that wraps the datastream(s) of which the asset consists. Such an XML document is named a Digital Item Declaration (DID); all LANL DIDs are compliant with the MPEG-21 DIDL specification. As such, for example, the different datastreams of the previously mentioned

scholarly article will be contained in a single DID, which will physically contain and/or reference the various datastreams that make up the complex object. The DID also contains information added by the ingestion process, for example, aimed at expressing relationships between contained datastreams, the media type of datastreams, etc. The actual use of DIDL in the LANL Repository, and the LANL DIDL profile is described in detail in [6] [note 3]. Appendix A shows an actual DID.

For the purpose of this paper, the use of identifiers in LANL DIDs is important. Two types of identifiers are distinguished. The manner in which they are conveyed in DIDs has been revised to some extent when compared to our previous D-Lib paper [6]:

- DID-identifiers: A DID-identifier is a globally unique identifier assigned by the repository ingestion process to each DID. The DID-identifier is conveyed in a ***Container***-level ***Descriptor*** that uses the `dii:identifier` element from the MPEG-21 DII XML Namespace [12]. During the ingestion process, DID ***Items*** and DID ***Components*** receive XML IDs [13] that are attached as attributes to the corresponding `didl:Item` and `didl:Component` XML elements from the MPEG-21 DIDL XML Namespace. As a result, they are addressable using a combination of the DID-identifier of the DID in which they are contained, and their own XML ID. DID-identifiers are crucial for the functioning of the LANL Repository itself and for its directly associated processes, such as the MPEG-21 DIP Engine that is discussed later. The DID-identifier and the associated XML IDs of a sample DID are shown in Appendix A.
- Content-identifiers: Content-identifiers are directly related to identifiers that are natively attached to a—set of—datastream(s) before their ingestion into the LANL Repository. They are typically inferred from the ingested assets. Indeed, in many cases those assets have identifiers that were associated with them when they were created or published. For example, Content-identifiers could be accession numbers for records in A&I databases, DOIs [note 4] for journal articles etc. According to the LANL DIDL profile [6], Content-identifiers are attached to the ***Item*** level of DIDs only. They are considered special metadata about the ***Item*** and are conveyed using the `dc:identifier` element from the DC XML Namespace [note 5] in an ***Item***-level ***Descriptor***. This is illustrated by the sample DID shown in Appendix A.

3. The multi-faceted use of the OAI-PMH in the LANL Repository

Once a delivered asset has been turned into a DID by the ingestion process, the DID is stored in an OAI-PMH repository. Data assets at LANL are typically received in large batches, since secondary or primary publishers that account for the bulk of the data to be stored in the LANL Repository deliver weekly or annual feeds. In those cases, an autonomous OAI-PMH repository is created per delivered batch. The OAI-PMH `identifier` used by these repositories is the DID-identifier, and the OAI-PMH `datestamp` is the datetime of creation or update of a DID. As those repositories only contain DIDs, the only supported `metadataPrefix` is DIDL. As a result, many autonomous OAI-PMH repositories exist in the LANL Repository. A special component, the Repository Index, keeps track of the creation and location of these autonomous OAI-PMH repositories. The Repository Index is itself an OAI-PMH repository. Through interaction with the Repository Index and the autonomous OAI-PMH repositories, harvesters working on behalf of downstream applications can selectively gather DIDs from the LANL Repository that were added or updated since the previous harvest, and—in good OAI-PMH Service Provider tradition—do something meaningful with them.

In order to relieve downstream harvesters from the burden of having to interact with all autonomous OAI-PMH repositories, and having to understand other details of the LANL Repository architecture, the OAI-PMH Federator is introduced. The OAI-PMH Federator exposes the whole LANL Repository as a single OAI-PMH repository. The OAI-PMH Federator becomes the single point of access to the LANL Repository for harvesters—each of which is internal to LANL—hiding the complexity of the LANL Repository environment from them. As a result, the applications on whose behalves the harvesters operate can permanently remain in sync with the LANL Repository situation using a uniform technique—OAI-PMH harvesting. The multi-faceted use of the OAI-PMH in the LANL Repository, as briefly summarized in this Section, will be explained in detail in a forthcoming paper.

4. A special Service Provider: the Identifier Resolver

Harvesters work on behalf of Service Providers, collect DIDs from the LANL Repository, and build services with the collected information. As a result, identifiers contained in the harvested DIDs become available in applications such as

search engines. As mentioned before, these identifiers can either be DID-identifiers identifying DIDs or DID entities, or Content-identifiers identifying content contained in DIDs. Obviously it is essential that, when such identifiers show up in downstream applications, the corresponding content can be retrieved from the LANL Repository. For this purpose the Identifier Resolver is introduced to the environment. The Identifier Resolver is a special-purpose Service Provider that collects the information it requires from the LANL Repository through recurrent OAI-PMH harvesting. From the harvested information, it only uses the DID-identifiers, the Content-identifiers, and the baseURL of the OAI-PMH repository in which these occur.

The Identifier Resolver consists of two modules, reflecting the nature of the contained identifiers:

- A DID-identifier module, shown in Table 1, which only contains DID-identifiers and the baseURL of the autonomous OAI-PMH repository in which the identified DID is contained.
- A Content-identifier module, shown in Table 2, which maps Content-identifiers to DID-identifiers.

Tables[1] and [2] illustrate the content of the Identifier Resolver. As can be seen, all identifiers are expressed using the proposed 'info' URI scheme [14] [note 6]. In Table 1, 'info:lanl-repo/i/58f202ac' and 'info:lanl-repo/i/002035b2' are DID-identifiers, of DIDs located in the OAI-PMH repository with `baseURL` "BaseURL(3)" and "BaseURL(6)", respectively. As these are DID-identifiers, the value of the OAI-PMH `identifiers` of the DIDs in these repositories is equal to the value of the DID-identifiers. In Table 2, Content-identifiers 'info:lanl-repo/apsmeta/123456', 'info:lanl-repo/aps/1202252' and 'info:lanl-repo/biosis/abcdef' identify content contained in the DIDs with DID-identifier 'info:lanl-repo/i/58f202ac' and 'info:lanl-repo/i/002035b2', respectively. Within these DIDs, the ***Items*** with which these Content-identifiers are associated have XML IDs '445998', '899952', and '444940', respectively. Appendix A shows the DID with DID-identifier 'info:lanl-repo/i/58f202ac'.

DID-Identifier	OAI-PMH repository
info:lanl-repo/i/58f202ac	BaseURL(3)
info:lanl-repo/i/002035b2	BaseURL(6)

Table 1: DID-identifier module of the Identifier Resolver

Content-identifier	DID-identifier	XML ID
info:lanl-repo/apsmeta/123456	info:lanl-repo/i/58f202ac	445998
info:lanl-repo/aps/1202252	info:lanl-repo/i/58f202ac	899952
info:lanl-repo/biosis/abcdef	info:lanl-repo/i/002035b2	444940

Table 2: Content-identifier module of the Identifier Resolver

The Identifier Resolver is accessible to applications in a number of ways, including the Handle protocol, a SOAP-based [15] access mechanism, and a C library. Through consultation of the Identifier Resolver, an application can use the OAI-PMH to retrieve the DID with a specified DID-identifier, or the DID in which content with a specified Content-identifier resides. This works as follows:

- If the object with identifier 'info:lanl-repo/i/58f202ac' is requested, a look-up in the Identifier Resolver will learn that it is located at BaseURL(3), in a DID with DID-identifier 'info:lanl-repo/i/58f202ac'. This DID can be obtained by issuing the OAI-PMH request `[BaseURL(3)?verb=GetRecord&identifier=info:lanl-repo/i/58f202ac&metadataPrefix=DIDL]`
- If the object with identifier 'info:lanl-repo/i/58f202ac#445998' is requested, a look-up of 'info:lanl-repo/i/58f202ac' in the Identifier Resolver will learn that it is located at BaseURL(3), in a DID with DID-identifier 'info:lanl-repo/i/58f202ac'. That DID can be obtained by issuing the OAI-PMH request `[BaseURL(3)?verb=GetRecord&identifier=info:lanl-repo/i/58f202ac&metadataPrefix=DIDL],` and from the resulting DID, the ***Item*** with XML ID '445998' can be extracted.
- If the object with identifier 'info:lanl-repo/biosis/abcdef' is requested, a look-up in the Identifier Resolver will learn that it is located at BaseURL(6), in a DID with DID-identifier 'info:lanl-repo/i/002035b2', and that—in this DID—it has the XML ID '444940'. From this information, the application can conclude that the requested object

is an asset contained in a DID. This DID can be obtained by issuing the OAI-PMH request `[BaseURL(6)?verb=GetRecord&identifier=info:lanl-repo/i/002035b2&metadataPrefix=DIDL]`, and from the resulting DID, the ***Item*** with XML ID '444940' can be extracted.

Because Content-identifiers can always be mapped to DID-identifiers through the Identifier Resolver, only DID-identifiers will be used for the discussions in the remainder of this paper.

In addition to the information shown in Table 2, the Identifier Resolver also keeps track of the location of assets for which information providers delivered updated versions. For DIDs, only a single version will ever correspond with a given DID-identifier. However, when an update for content contained in a DID is delivered, typically a new DID is created and stored in another OAI-PMH repository as was the original DID. As a result, the location information for that content changes, and this is reflected in the Identifier Resolver by updating the content of the location information for the Content-identifier involved and saving previously existing information in a history section.

5. Disseminating assets from the LANL Repository

As was described, downstream applications obtain DIDs from the LANL Repository through OAI-PMH harvesting. These applications will reference DIDs, DID entities, or more commonly, assets contained in DIDs to allow agents that use the applications to retrieve the referenced entities from the LANL Repository. In order to facilitate the retrieval of various —dynamically generated—disseminations of stored assets, a solution has been designed that is based on the MPEG-21 DIP specification, and on the NISO OpenURL. Various aspects of the solution, of which an initial implementation has been developed, are described in this section.

5.1 Associating Services with DID Entities: MPEG-21 Digital Item Methods

MPEG-21 Digital Item Processing (DIP) specifies an architecture pertaining to the dissemination of DIDs. This MPEG-21 Part, that remains to be standardized, introduces the concept of a Digital Item Method (DIM), as a means to associate services with a DID and/or its entities. It is worthwhile mentioning that the MPEG-21 DIM concept is closely related to Fedora's "behavior" concept [4,5]. A DIM is physically contained in the same DID as the entity of the DIDL data model with which it is associated. In the current, pre-standard, practice a DIM is typically accommodated in a ***Component*** entity of the DID, which must contain at least 2 other DID entities:

- A ***Resource*** that contains or references the actual method. In MPEG-21 DIP, the DIM code is expressed using a language that is closely related to the ECMAScript Language [16]. As will be explained in the section on the DIP Engine, this ECMA Script does not contain all the actual code required to implement the service. Rather, the ECMA Script is used to bootstrap and coordinate the implementation of the service. It contains calls to code that does the real work.
- A ***Descriptor*** that contains a sequence of `dip:Argument` elements from the DIP XML Namespace, one per argument of the DIM. The value of a `dip:Argument` element always identifies a DID entity.

A DIM is associated with an entity of a DID using the `dip:objectType`/`dip:Argument` technique, explained in detail in [6]:

- Using a special-purpose ***Descriptor***, a DID entity can be accorded a `dip:ObjectType` element from the DIP XML Namespace.
- Using another special-purpose ***Descriptor***, a DIM can be accorded a `dip:Argument` element, also from the DIP XML Namespace.
- When, in a DID, the value of a `dip:ObjectType` element of a DID entity, and the value of the `dip:Argument` of a DIM are equal, then the DIM can be applied to the DID entity

This is illustrated in the DID shown in Appendix A, in which a service is associated with a ***Component*** through the value 'urn:uuid:8f64eabf-1dd2-11b2-a3f1-0800209a5b6b'. That value is used as the content of both the `dip:ObjectType` element of the ***Component***, and the `dip:Argument` element of the DIM. The DIP specification allows entities to have more than one `dip:ObjectType`. Also, a DIM can bind to more than one entity by using multiple `dip:Argument` elements, each of which connects via the `dip:ObjectType` to the entities. It is worthwhile mentioning that changes have been made to the DIP specification since the publication of our previous D-Lib paper

[6]. Processing methods in DIDs are now recognizable by their use of the reserved `dip:Arguments` element. The notion of a special ***Item***—the ***Processing Item***—reserved to contain processing methods is no longer supported.

5.2 Dynamically Associating Services with DID entities in the LANL Repository: the Placeholders and the DIP Table

As explained in [6], the authors felt a general level of discomfort with statically embedding DIMs in DIDs stored in the LANL Repository. Therefore, it was decided to embed Placeholders for DIMs instead of actual DIMs in stored DIDs. When the dissemination of a stored DID is requested, the DIM Inserter module dynamically adds DIMs to that DID in a process that is based on the values of the Placeholders contained in the DID. The result is a Completed DID, which contains assets and associated methods.

Placeholders can be attached at the ***Container***, ***Item*** and ***Component*** level of a DID. At the ***Container*** level, the Placeholder conveys to which 'family' the DID belongs. For example, the Placeholder value 'container:ai' shown in Table 3 is used to convey that a DID only contains metadata. This 'family' information is closely related to the ingestion process. At the ***Component*** level, the Placeholder specifies the digital format of the contained datastream. Awaiting the emergence of global unique identifiers for digital formats, values are currently used as shown in the second entry of Table 3. These values convey the media type of a datastream as well as a broad classification of the nature of its format. The latter is based on an ontology under development by the Digital Format Registry project [17]. The exact nature of ***Item***-level Placeholders is a matter of ongoing research. Placeholder values are conveyed in DIDs via a `diph:PlaceHolder` element from a self-defined XML Namespace.

The actual insertion of DIMs in a DID is achieved through a look-up in a special-purpose registry—the DIP Table. The DIP Table lists all services that can be associated with DID entities from the LANL Repository. In the DIP Table, each service has a Service Identifier. Also, for each service, the DIP Table contains a Placeholder value—as used in the DIDs—with which the service is associated. For example, the first entry of [Table 3] shows that the service with Service Identifier 'info:lanl-repo/service/table_of_contents' is associated with all DID entities that have 'container:ai' or 'container:aps' as a Placeholder value. For each service, the DIP Table also lists a pointer to the DIM code that actually implements it.

Service Identifier	info:lanl-repo/service/table_of_contents
Placeholder value	container:ai
Pointer to DIM Code	gov.lanl.library.dip.toc
Description	*Container level service that displays a Table of Contents of a DID as an XHTML page, listing all contained entities as well as the services that are available for them*
Service Identifier	info:lanl-repo/service/table_of_contents
Placeholder value	container:aps
Pointer to DIM Code	gov.lanl.library.dip.toc
Description	*Container level service that displays a Table of Contents of a DID as an XHTML page, listing all contained entities as well as the services that are available for them*
Service Identifier	info:lanl-repo/service/marc_2_mods

Placeholder value	component:content-stream:text:structured-text:mark-up-language:xml#application/marc+xml
Pointer to DIM Code	gov.lanl.library.dip.marctomods
Description	*Component level service that disseminates stored MARCXML as MODS*

Table 3: Three entries of the DIP Table

The dynamic insertion of DIMs in a DID, as performed by the DIM Inserter module of the LANL Repository, is explained by means of the following example. It is also illustrated in the left part of [Figure 2]. Consider a DID, which has a ***Component***-level Placeholder with a value of 'component:content-stream:text:structured-text:mark-up-language:xml#application/marc+xml'. A look-up of this Placeholder value in the DIP Table reveals that a service with Service Identifier 'info:lanl-repo/service/marc_2_mods', and a pointer to DIM code 'gov.lanl.library.dip.marctomods' are associated with it. This service must be added to the DID. This requires:

- Dereferencing the pointer to the DIM code, and inserting the actual code as a ***Component*** in the DID. Hereby, the Service Identifier 'info:lanl-repo/service/marc_2_mods' is also inserted as the identifier of the ***Component***.
- Associating the inserted DIM with the ***Component*** which had 'component:content-stream:text:structured-text:mark-up-language:xml#application/marc+xml' as the Placeholder value. This is achieved by inserting a `dip:objectType` element, and associating it with the DIM using the `dip:ObjectType/dip:Argument` technique of MPEG-21 DIP described above.

In order to embed all relevant DIMs into a DID upon dissemination, the process as explained above is repeated for every Placeholder entry found in the DID. The DID of Appendix A actually illustrates the result of the above actions applied for Placeholder values 'container:aps' and 'component:content-stream:text:structured-text:mark-up-language:xml#application/marc+xml'.

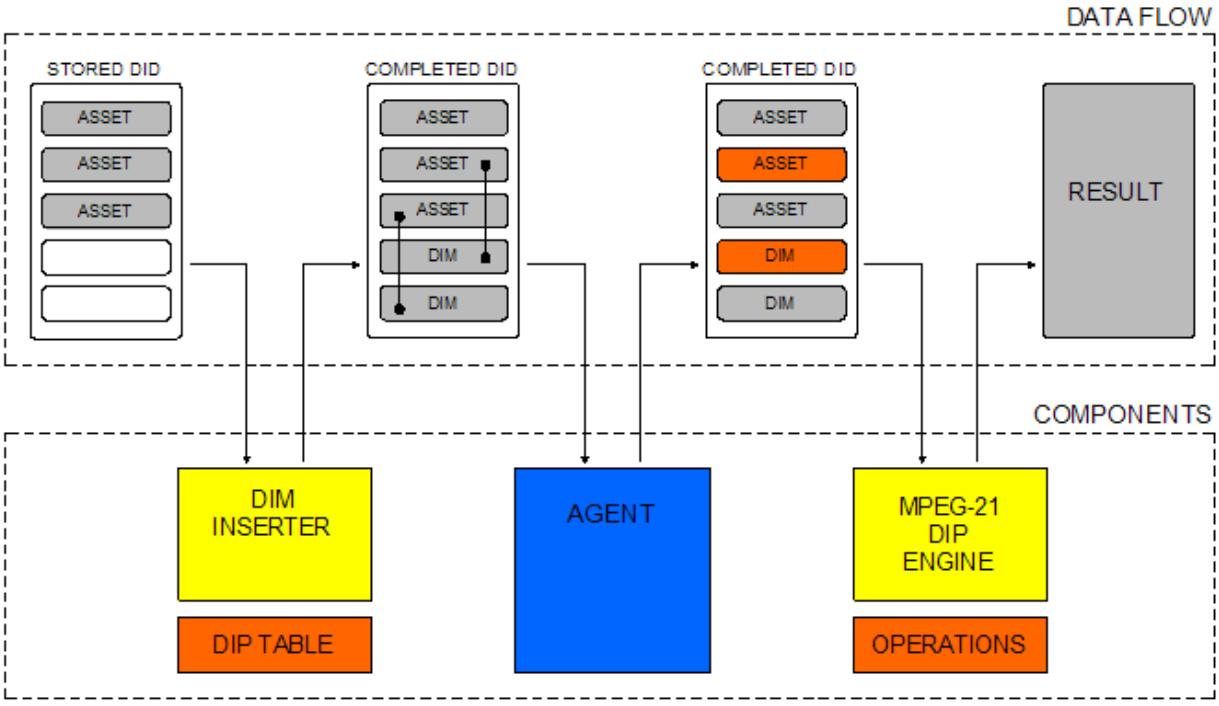

Figure 2: Dynamic dissemination of stored DIDs

5.3 Using an MPEG-21 DIP Engine to process service requests for DIDs

MPEG-21 DIP introduces a component, named a DIP Engine, that is capable of processing DIDs upon request of an

agent. A DIP Engine is able to respond to service requests in which the following information is conveyed:

- An actual DID.
- An identification of the entity of the DID for which the service is requested.
- An identification of the method—DIM—contained in the DID that implements the service.

The actual functioning of the LANL DIP Engine is illustrated in Figure 2. Upon receipt of a service request, the DIP Engine will extract the identified DID entity, and DIM. As was mentioned, DIMs are expressed using a language that normatively includes ECMAScript. This ECMAScript does not actually contain all the code required for the implementation of the service request. Rather, the ECMAScript is used by the DIP Engine to bootstrap the process of implementing the service request and to actually coordinate it. To that end, the ECMAScript contains calls to so-called Digital Item Operations. Two types of Digital Item Operations are distinguished in MPEG-21 DIP:

- Digital Item Base Operations—DIBOs: DIBOs are Digital Item Operations that must be supported by every compliant DIP Engine. Currently defined DIBOs include operations to manipulate DIDs at the XML level—e.g. `GetDIDLNode` —and operations that have general application across a wide range of domains, applications and media types—e.g. `PlayResource`.
- Digital Item extension Operations—DIxOs: DIxOs are Digital Item Operations that are not defined by MPEG-21 DIP, but are rather defined at the level of specific communities, and applications. DIxOs are the extensibility mechanism built into the DIP Engine.

An MPEG-21 DIP Engine is typically thought of as a client-side software component that operates on a user terminal. At LANL, a server-side prototype implementation of a DIP Engine has been developed. It currently supports only a few of the DIBOs defined by MPEG-21. But various DIxOs have been coded that implement services that are relevant for and specific to the nature of the collection stored in the LANL Repository. As a matter of fact, all services listed in Table 3 are implemented using DIxOs. Also, Java-based templates have been developed to facilitate the straightforward creation of more DIxOs. For example, one template can be used as the basis for all DIxOs that use XSL transforms, another for DIxOs that need to call external applications, and yet another for the interaction with Web Services.

5.4 Using NISO OpenURL to convey service requests

Through OAI-PMH harvesting, downstream applications obtain DIDs and use contained assets in their services. For example, a search engine might extract all textual information from harvested DIDs and make it available for searching by end-users. In this case, brief search results will point back to the corresponding asset stored in the LANL Repository. Obviously, in such pointers the DID-identifiers, their associated XML IDs, and the Content-identifiers will play a crucial role. Also, as was shown, the proposed solution allows for the definition of several services aimed at delivering various disseminations of content. Hence, requests to retrieve content should also be able to convey which actual dissemination of the identified content is requested. This, as a matter of fact, is equivalent to conveying the Service Identifier, as shown in Table 3, of the service that implements the required dissemination. Since many downstream applications are Web-based, the discussion will assume that requesting disseminations from the LANL Repository is an HTTP-based process.

It turns out that the forthcoming NISO OpenURL Standard [10] provides a perfect framework to convey such dissemination requests. The initial OpenURL specification [18] was specifically introduced for the purpose of reference linking, and was targeted at facilitating the provision of context-sensitive reference links for popular types of scholarly works such as journal articles and books [19]. Hereby, identifiers and metadata describing the work are conveyed using a controlled-vocabulary HTTP GET request to a user-specific linking server, which uses its knowledge base and the incoming information to provide a user with appropriate services pertaining to the work. A generalization of the essential components of the initial OpenURL solution [20] inspired the nature of the forthcoming NISO OpenURL Standard, which provides a generic framework for the delivery of context-sensitive services pertaining to whichever type of resource referenced in a networked environment. To that end, the OpenURL Standard introduces the notion of a ContextObject, which is an information construct that contains descriptions of various entities involved in the process of providing context-sensitive services. The entities are shown in Table 4.

Entity	Definition
Referent	The entity about which the ContextObject was created—the referenced resource
ReferringEntity	The entity that references the Referent
Requester	The entity that requests services pertaining to the Referent
ServiceType	The entity that defines the type of service requested
Resolver	The entity at which a request for services is targeted
Referrer	The entity that generated the ContextObject

Table 4: The NISO OpenURL ContextObject

Each entity of the ContextObject can be described by means of identifiers, by means of metadata, and/or by means of private data. A ContextObject can be represented in many ways, and currently, a Key/Encoded-Value (KEV) representation and an XML representation have been defined. A representation of a ContextObject can be transported to a networked system named a Resolver, in order to request services pertaining to the Referent described in it. To decide upon the nature of such services, the Resolver may take entities other than the Referent into account. The network transport can occur using various network protocols, and currently transport over HTTP and HTTPS have been defined.

From the above, two mappings follow easily:

- The asset from the LANL Repository for which a dissemination is requested ~ the Referent of the OpenURL ContextObject.
- The type dissemination that is requested for the asset ~ the ServiceType of the OpenURL ContextObject.

An OpenURL Resolver is introduced in the LANL Repository to which all dissemination requests originating from downstream applications will be targeted. Using examples provided earlier in this paper (Tables 1-3), the HTTP transport of the OpenURL Standard, and its KEV representation of the ContextObject, the following are valid OpenURLs to convey dissemination requests to this OpenURL Resolver:

- Display a Table of Contents for the DID with DID-identifier 'info:lanl-repo/i/58f202ac':
```
BaseURL(OpenURL Resolver)?
    url_ver=Z39.88-2004&
    rft_id=info:lanl-repo/i/58f202ac&
    svc_id=info:lanl-repo/service/table_of_contents
```
- Display the MARCXML [note 7] record with DID-identifier 'info:lanl-repo/i/58f202ac#445998C1' as MODS [note 8]:
```
BaseURL(OpenURL Resolver)?
    url_ver=Z39.88-2004&
    rft_id=info:lanl-repo/i/58f202ac#445998C1&
    svc_id=info:lanl-repo/service/marc_2_mods
```

Content-identifiers can also be used in OpenURL requests. However, as Content-identifiers can always be mapped to a corresponding DID-identifier (see Section 4), only DID-identifiers are considered here. Currently, the Referent and ServiceType are the only entities of the ContextObject that are used in OpenURL requests. But, NISO OpenURL allows expressing other entities that can be taken into account when responding to dissemination requests. Conveying Requester information may be of particular interest, as this would allow adapting the actual dissemination to the agent requesting it. Requester information could convey identity, and this would allow responding differently to the same service request for the same asset depending on whether the requesting agent is human or machine. Or different humans could receive different disseminations based on recorded preferences. But the NISO OpenURL specification is purposely very generic and extensible, and would also support conveying the characteristics of a user's terminal, and/or the user's location via the Requester entity. Such information could be passed on to the DIP Engine and could be taken into account in the delivery of an actual dissemination. As a matter of fact, the expressiveness of NISO OpenURL seems to resonate nicely with the nature of the MPEG-21 Digital Item Adaptation (DIA) [21] effort that focuses on

transforming assets for delivery by taking into account characteristics of the requesting terminal, network conditions and other contextual information.

5.5 Putting MPEG-21 DID, MPEG-21 DIP, NISO OpenURL, and the OAI-PMH together

This section provides a walkthrough of the processes involved in responding to the two OpenURL requests shown above.

The Table of Contents request involves the following steps:

1. Through interaction with the Identifier Resolver, the OpenURL Resolver learns that the Referent with identifier 'info:lanl-repo/i/58f202ac' is a DID located in the OAI-PMH repository with `baseURL` "BaseURL(3)" (see Table 1).
2. The OpenURL Resolver retrieves this DID from its OAI-PMH repository by issuing an OAI-PMH `GetRecord` request `[BaseURL(3)?verb=GetRecord&identifier=info:lanl-repo/i/58f202ac&metadataPrefix=DIDL]`.
3. The OpenURL Resolver passes the retrieved DID on to the DIM Inserter, which dynamically adds DIMs to the DID based on contained Placeholder values and look-ups in the DIP Table.
4. The DIM Inserter returns the completed DID (see Appendix A) to the OpenURL Resolver.
5. The OpenURL Resolver passes the completed DID, the identifier of the Referent, and the identifier of the ServiceType to the DIP Engine. As a matter of fact, these identifiers are passed on to the DIP Engine as XPath expressions that result from locating the respective identifiers in the DID. However, to avoid complexity this example continues using the actual identifiers.
6. Through inspection of the DID, the DIP Engine determines whether the requested service with identifier 'info:lanl-repo/service/table_of_contents' can be applied to the DID entity with identifier 'info:lanl-repo/i/58f202ac'. This is achieved by checking the `dip:ObjectType/dip:Argument` correspondency.
7. Assuming that the service can indeed be applied, the appropriate DIM is extracted from the DID and is executed by the DIP Engine. This DIM calls a DIxO that actually transforms the XML DID to the XHTML Table of Contents.
8. The DIP Engine returns the XHTML Table of Contents to the OpenURL Resolver.

With regard to the MARCXML to MODS request, the process involved in responding to this OpenURL request is similar to the one described above, with these nuances:

1. The Referent identifier 'info:lanl-repo/i/58f202ac#445998C1' reveals to the OpenURL Resolver that the Referent is a DID entity with XML ID '445998C1', contained in the DID with DID-identifier 'info:lanl-repo/i/58f202ac'. This DID can be retrieved as was described in Steps 1 and 2 above.
2. The remainder of the process is as described in Steps 3 through 8 above. As this example is concerned with a DID entity rather than a complete DID, the DIxO involved will operate on an extraction of the DID rather than on the complete DID.

6. Conclusion

This paper has described aspects of the LANL Repository architecture pertaining to the dissemination of complex objects or their contained assets. In this architecture, NISO OpenURL is used to encode dissemination requests. This takes the actual use of OpenURL to a new level. Indeed, OpenURL has so far only been used to deliver services that were calculated based on metadata that describes a referenced resource. In the proposed solution, the services are based on the resource itself. As NISO OpenURL is generic and extensible, it offers a flexible framework for the expression of contextual dissemination requests. Although the authors have only scratched the surface of the use of NISO OpenURL in the proposed solution, it seems fair to state that it resonates nicely with MPEG-21 DIP, and potentially MPEG-21 DIA.

NISO OpenURL is also attractive in another sense. One of the core assumptions made in the OpenURL Framework is the existence of a multitude of OpenURL Resolvers, each of which can have its own behavior when responding to OpenURL requests. Typically this behavior is based on the nature and/or preferences of a community on whose behalf

a Resolver operates. The natively distributed and protocol-based nature of the proposed architecture allows us to speculate on its deployment in a Web context. What was described as the LANL Repository now becomes a federation of OAI-PMH repositories distributed across the Web. In such a federation, components such as the Repository Index, the Identifier Resolver and the OAI-PMH Federator can play roles that are very similar to those described in the context of the LANL Repository. One can easily imagine the existence of an OpenURL Resolver per institution in the federation. As is the case in the OpenURL Framework that has successfully been deployed for reference linking in the scholarly community, users would direct dissemination requests to the OpenURL Resolver of their institution. The institutional Resolver would respond to such requests by obtaining a stored complex object from the appropriate OAI-PMH repository in the federation. And, in a process similar to the one performed by the described DIM Inserter, locally defined services could be attached to the object. Actually, assuming the existence of globally unique identifiers for digital formats, the addition of such services based on digital format is not even difficult to imagine. The result is a framework in which each institution of the federation can implement different preferences with respect to the nature of the dissemination of the same content.

Acknowledgements

The authors would like to thank their colleagues Henry Jerez, Xiaoming Liu, and Thorsten Schwander from the LANL Digital Library Research and Prototyping team for their contributions to the reported work, and the LANL library director, Rick Luce, for his ongoing support. Jeroen Bekaert also wishes to thank the Fund for Scientific Research (Flanders, Belgium) for his PhD scholarship.

References

1. Bollen, J. and Luce, R. "Evaluation of Digital Library Impact and User Communities by Analysis of Usage Patterns," D-Lib Magazine, Volume 8, Number 6, June 2002. *http://dx.doi.org/ 10.1045/june2002-bollen*

2. Van de Sompel, H., Young J. and Hickey T. "Using the OAI-PMH ... Differently," D-Lib Magazine, Volume 9, Number 7/8, July/August 2003. *http://dx.doi.org/10.1045/july2003-young*

3. Smith M. et al. "DSpace An Open Source Dynamic Digital Repository", D-Lib Magazine, Volume 9, Number 1, January 2003. *http://dx.doi.org/10.1045/january2003-smith*

4. Payette, S. and Lagoze, C. "Flexible and Extensible Digital Object and Repository Architecture," Second European Conference on Research and Advanced Technology for Digital Libraries, Heraklion, September 1998. *http://www.cs.cornell.edu/payette/papers/ECDL98/FEDORA.html*

5. Staples, T., Wayland, W. and Payette S. "The Fedora Project: An Open-Source Digital Object Repository System," D-Lib Magazine, April 2003. *http://dx.doi.org/10.1045/april2003-staples*

6. Bekaert, J., Hochstenbach, P. and Van de Sompel, H. "Using MPEG-21 DIDL to Represent Complex Digital Objects in the Los Alamos National Laboratory Digital Library," D-Lib Magazine, Volume 9, Number 11, November 2003. *http://dx.doi.org/10.1045/november2003-bekaert*.

7. Lagoze, C., Van de Sompel, H., Nelson, M. and Warner, S. "The Open Archives Initiative Protocol for Metadata Harvesting - Version 2.0," 2003. *http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm*

8. MPEG-21, Information Technology, Multimedia Framework, "Part 2: Digital Item Declaration," *ISO/IEC 21000-2:2003*, March 2003.

9. MPEG-21, Information Technology, Multimedia Framework, "Part 10: MPEG-21 Digital Item Processing," *ISO/IEC JTC1/SC29/WG11, N6173*, Hawaii, December 2003.

10. NISO committee AX. "The OpenURL Framework for Context-Sensitive Services," *ANSI/NISO Z39.88-2004*, November 2003. *http://library.caltech.edu/openurl/StandardDocuments/Part1-Ballot-20031111.pdf*

11. Sun, S., et al. "Handle System Overview. Internet Engineering Task Force (IETF) Request for Comments (RFC), RFC 3650", November 2003. *http://hdl.handle.net/4263537/4069*

12. MPEG-21, Information Technology, Multimedia Framework , "Part 3: Digital Item Identification," *ISO/IEC 21000-3:2003*, March 2003.

13. W3C, "Proposal for XML Fragment Identifier Syntax 0.9," Working Group Note, Sept 2003. *http://www.w3.org/TR/xml-fragid/*

14. Van de Sompel, H., Hammond, T., Neylon, E., and Weibel S.L. "The "info" URI Scheme for Information Assets with Identifiers in Public Namespaces", December 2003. *http://www.ietf.org/internet-drafts/draft-vandesompel-info-uri-01.txt*

15. W3C, "SOAP Version 1.2 Part 1: Messaging Framework", Recommendation, June 2003. *http://www.w3.org/TR/SOAP/*

16. ECMA-262, "ECMAScript Language Specification, 3rd Edition," December 1999. *http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf*

17. Abrams, S. L. and Seaman, D. "Towards a global digital format registry," World Library and Information Congress: 69th IFLA General Conference and Council, Berlin, August 1-9, 2003. *http://www.ifla.org/IV/ifla69/papers/128e-Abrams_Seaman.pdf*

18. Van de Sompel, H., Hochstenbach, P. and Beit-Arie, O. "OpenURL Syntax Description," 2000. *http://www.openurl.info/registry/docs/pdf/openurl-01.pdf*

19. Van de Sompel, H. and Beit-Arie, O. "Open Linking in the Scholarly Information Environment Using the OpenURL Framework," D-Lib Magazine, Volume 7, Number 3, March 2001. *http://dx.doi.org/10.1045/march2001-vandesompel*

20. Van de Sompel, H. and Beit-Arie, O. "Generalizing the OpenURL Framework beyond References to Scholarly Works," D-Lib Magazine, Volume 7, Number 3, March 2001. *http://dx.doi.org/10.1045/july2001-vandesompel*

21. MPEG-21, Information Technology, Multimedia Framework, "Part 2: Digital Item Adaptation," *ISO/IEC JTC1/SC29/WG11, N5845*, Trondheim, July 2003.

Notes

1. MPEG-21 DIDL Working Draft: *http://xml.coverpages.org/MPEG21-WG-11-N3971-200103.pdf*
2. MPEG-21 DIDL Working Draft Schema: *http://download.webct.com/public/ims/2.0/MPEG21.xsd*
3. LANL DID: *http://lib-www.lanl.gov/proto/MPEG-21/*
4. DOI: *http://www.doi.org/*
5. Dublin Core Metadata Initiative, Dublin Core, *http://dublincore.org/*
6. info URI Registry: *http://info-uri.info/registry/*
7. MARC 21 XML Schema, *http://www.loc.gov/standards/marcxml/*
8. Metadata Object Description Schema, *http://www.loc.gov/standards/mods/*

Appendix A: A DID containing DIMs associated with a DID entity

```
<?xml version="1.0" encoding="UTF-8"?>
<didl:DIDL xmlns:didl="urn:mpeg:mpeg21:2002:02-DIDL-NS">
    <didl:Container>
        <!-- DID-identifier -->
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml; charset=UTF-8">
                <dii:Identifier xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS">
                    info:lanl-repo/i/58f202ac</dii:Identifier>
            </didl:Statement>
        </didl:Descriptor>
        <!-- Container-level Placeholder -->
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml; charset=UTF-8">
                <diph:PlaceHolder
                    xmlns:diph="http://library.lanl.gov/2003-09/STB-RL/DIPH">
                    container:aps</diph:PlaceHolder>
            </didl:Statement>
        </didl:Descriptor>
        <!-- ObjectType of the Container, added by DIM Inserter -->
        <!-- Corresponds with Argument of DIM (Table of Contents) below -->
        <didl:Descriptor>
                <didl:Statement mimeType="text/xml; charset=UTF-8">
                    <dip:ObjectType xmlns:dip="urn:mpeg:mpeg21:2002:01-DIP-NS">

urn:uuid:58f202ac-22cf-11d1-b12d-002035b29092</dip:ObjectType>
                </didl:Statement>
        </didl:Descriptor>
        <!-- Item containing a MARCXML metadata record -->
        <!-- DID-identifier - XML ID of the Item -->
        <didl:Item id="445998">
            <!-- Content-identifier of the Item -->
            <didl:Descriptor>
                <didl:Statement mimeType="text/xml; charset=UTF-8">
                  <diadm:Admin
xmlns:diadm"=http://libary.lanl.gov/2003-11/STB-RL/DIADM">
                      <dc:identifier xmlns:dc="http://purl.org/dc/elements/1.1/">
                          info:lanl-repo/apsmeta/123456</dc:identifier>
                  <diadm:Admin>
                </didl:Statement>
            </didl:Descriptor>
            <!-- Component containing the MARCXML datastream -->
            <!-- DID-identifier - XML ID of the Component -->
            <didl:Component id="445998C1">
                <!-- Component-level Placeholder -->
                <didl:Descriptor>
                    <didl:Statement mimeType="text/xml; charset=UTF-8">
                        <diph:PlaceHolder

xmlns:diph="http://library.lanl.gov/2003-09/STB-RL/DIPH">
                            component:content-stream:text:structured-text:mark-up-
                            language:xml#application/marc+xml</diph:PlaceHolder>
                    </didl:Statement>
                </didl:Descriptor>
                <!-- ObjectType of the datastream, added by DIM Inserter -->
                <!-- Corresponds with Argument of DIM (MARCXML to MODS) below -->
                <didl:Descriptor>
                    <didl:Statement mimeType="text/xml; charset=UTF-8">
                        <dip:ObjectType xmlns:dip="urn:mpeg:mpeg21:2002:01-DIP-NS">

urn:uuid:8f64eabf-1dd2-11b2-a3f1-0800209a5b6b</dip:ObjectType>
                    </didl:Statement>
```

```
                </didl:Descriptor>
                <!-- The actual MARCXML datastream -->
                <didl:Resource mimeType="text/xml; charset=UTF-8">
                    <record xmlns="http://www.loc.gov/MARC21/slim">
                        <leader>01142cam 2200301 a 4500</leader>
                        <controlfield tag="005">19930521155141.9</controlfield>
                        <datafield tag="010" ind1=" " ind2=" ">
                            <subfield code="a">92005291</subfield>
                        </datafield>
                        ...
                </didl:Resource>
            </didl:Component>
        </didl:Item>
        <!-- Item containing a full-text document -->
        <!-- DID-identifier - XML ID of the Item -->
        <didl:Item id="899952">
            <!-- Content-identifier of the Item -->
            <didl:Descriptor>
                <didl:Statement mimeType="text/xml; charset=UTF-8">
                  <diadm:Admin
xmlns:diadm"=http://libary.lanl.gov/2003-11/STB-RL/DIADM">
                      <dc:identifier xmlns:dc="http://purl.org/dc/elements/1.1/">
                          info:lanl-repo/aps/1202252</dc:identifier>
                  <diadm:Admin>
                </didl:Statement>
            </didl:Descriptor>
            <!-- Component containing the full-text datastream -->
            <!-- DID-identifier - XML ID of the Component -->
            <didl:Component id="899952C1">
                <!-- Component-level Placeholder -->
                <didl:Descriptor>
                    <didl:Statement mimeType="text/xml; charset=UTF-8">
                        <diph:PlaceHolder

xmlns:diph="http://library.lanl.gov/2003-09/STB-RL/DIPH">
                            component:content-stream:image:still:page-
                            description:pdf#application/pdf</diph:PlaceHolder>
                    </didl:Statement>
                </didl:Descriptor>
                <!-- The actual full-text datastream -->
                <didl:Resource encoding="base64" mimeType="application/pdf">
                    PSJjIj5jMTk5My48L3N1YmZpZWxkPg0KICAgIDw9uIHhtbG5zSJodHgKICAgIDxk
                    dGFnPSIzMDAiIGluZDE9IiAiIGluZDI9IiAiPg0KICAgICAgPHN1YmZpZWxkIGNv
                    cmVzdG9yZWQgdG8g…
                <didl:Resource>
            </didl:Component>
        </didl:Item>
        <!-- Item containing the DIM that implements the MARCXML to MODS service -->
        <!-- Inserted by the DIM Inserter after lookup of Placeholder in DIP Table
-->
        <didl:Item id="748083">
            <!-- identification of the DIM -->
            <didl:Descriptor>
                <didl:Statement mimeType="text/xml; charset=UTF-8">
                  <diadm:Admin
xmlns:diadm="http://libary.lanl.gov/2003-11/STB-RL/DIADM">
                      <dc:Identifier xmlns:dc="http://purl.org/dc/elements/1.1/">
                          info:lanl-repo/service/marc_2_mods</dc:Identifier>
                  <diadm:Admin>
                </didl:Statement>
            </didl:Descriptor>
            <!-- Actual DIM -->
            <!-- Obtained from look-up of Placeholder value
                component:content-stream:text:structured-text:mark-up-language:xml#
                application/marc+xml -->
```

```
            <didl:Component id="748083C1">
                <!-- Argument of the DIM -->
                <!--Corresponds with ObjectType attached to MARCXML -->
                <didl:Descriptor>
                    <didl:Statement mimeType="text/xml; charset=UTF-8">
                        <dip:Arguments xmlns:dip="urn:mpeg:mpeg21:2002:01-DIP-NS">
                            <dip:Argument>

urn:uuid:8f64eabf-1dd2-11b2-a3f1-0800209a5b6b</dip:Argument>
                        </dip:Arguments>
                    </didl:Statement>
                </didl:Descriptor>
                <!-- DIM ECMAScript corresponding with DIM Pointer
                    gov.lanl.lib.dip.marctomods -->
                <didl:Resource mimeType="application/DIM">
                ...
                </didl:Resource>
            </didl:Component>
        </didl:Item>
        <!-- Item containing the DIM that implements the Table of Contents service
-->
        <!-- Inserted by the DIM Inserter after lookup of Placeholder in DIP Table
-->
        <didl:Item id="539064">
            <!-- identification of the DIM -->
            <didl:Descriptor>
                <didl:Statement mimeType="text/xml; charset=UTF-8">
                  <diadm:Admin
xmlns:diadm="http://libary.lanl.gov/2003-11/STB-RL/DIADM">
                    <dc:Identifier xmlns:dc="http://purl.org/dc/elements/1.1/">
                        info:lanl-repo/service/table_of_contents</dc:Identifier>
                  <diadm:Admin>
                </didl:Statement>
            </didl:Descriptor>
            <!-- Actual DIM -->
            <!-- Obtained from look-up of Placeholder value container:aps -->
            <didl:Component id="539064C1">
                <!-- Argument of the DIM -->
                <!--Corresponds with ObjectType attached to Container -->
                <didl:Descriptor>
                    <didl:Statement mimeType="text/xml; charset=UTF-8">
                        <dip:Arguments xmlns:dip="urn:mpeg:mpeg21:2002:01-DIP-NS">
                            <dip:Argument>

urn:uuid:58f202ac-22cf-11d1-b12d-002035b29092</dip:Argument>
                        </dip:Arguments>
                    </didl:Statement>
                </didl:Descriptor>
                <!-- DIM ECMAScript corresponding with DIM Pointer
                    gov.lanl.library.dip.toc -->
                <didl:Resource mimeType="application/DIM">
                ...
                </didl:Resource>
            </didl:Component>
        </didl:Item>
    </didl:Container>
</didl:DIDL>
```

DOI: 10.1045/february2004-bekaert