Core DOI Specification

Version 1.0   Approved by IDF Board 15 June 2005

Abstract

The Digital Object Identifier (DOI®) is a system for implementing an unambiguous alphanumeric string, or identifier that references an intellectual property entity. The syntax of the identifier numbering scheme is such that any identifier can be expressed in a form suitable for use with the DOI system.

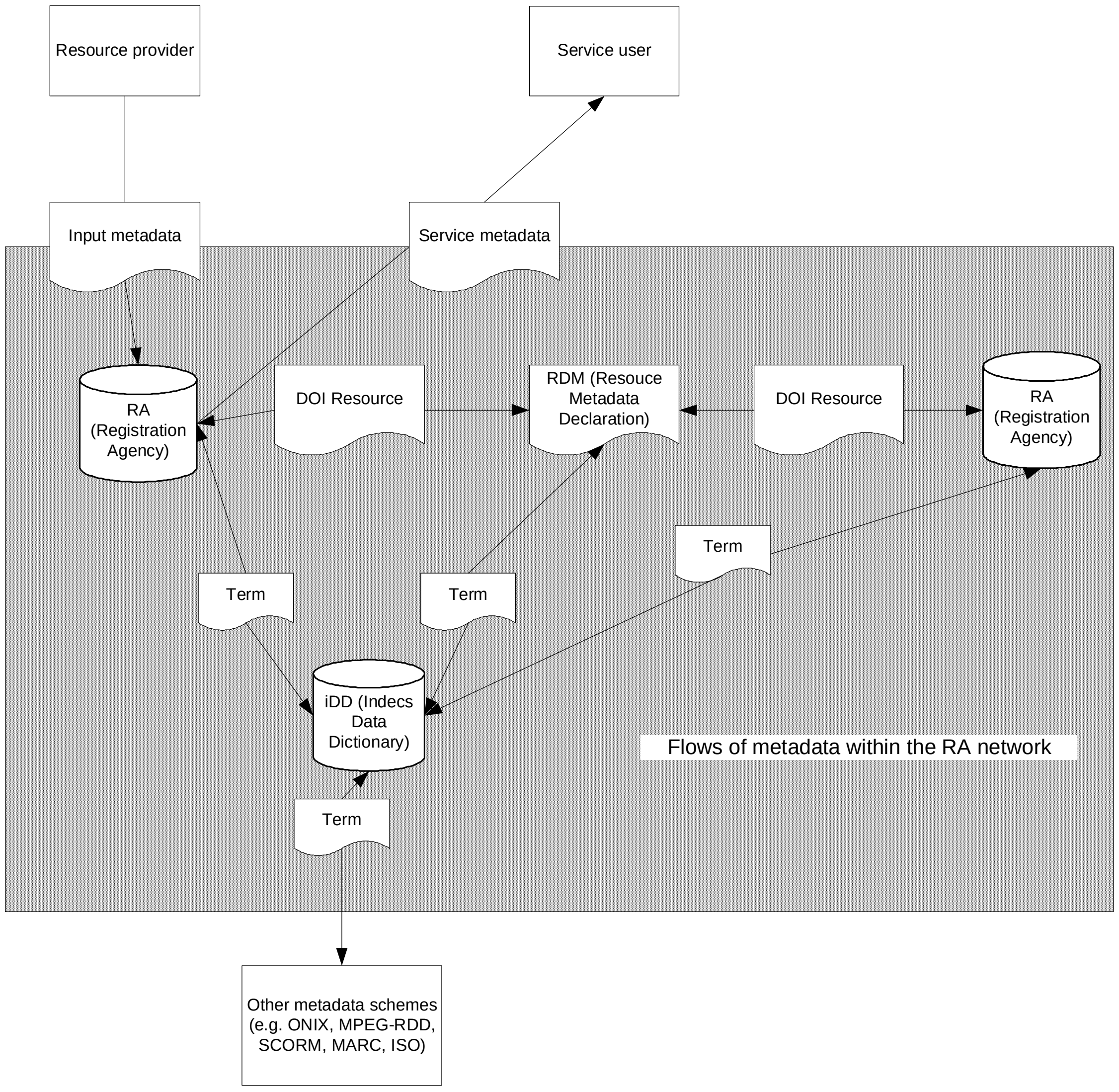

Resource provider
Service user
Input metadata
Service metadata
RA (Registration Agency)
DOI Resource
RDM (Resouce Metadata Declaration)
DOI Resource
RA (Registration Agency)
Term
Term
Term
iDD (Indecs Data Dictionary)
Flows of metadata within the RA network
Term
Other metadata schemes (e.g. ONIX, MPEG-RDD, SCORM, MARC, ISO)

Figure 1 Top-level view of a DOI system

Outline:

Part A. Definition of a DOI system

In the following description, the term "object" generally refers to an identifiable instance of intellectual property.

The DOI system is, at its core, a system for implementing labels. It has the following notable features:

- DOI is a persistent identifier
  - A DOI differs from commonly used Internet pointers to material such as the URL. A URL, when interpreted, refers to a particular IP address where information specified, in part, by the URL can be retrieved. A DOI is a reference to metadata about an object. This metadata serves to identify the object. Thus, a DOI can be viewed as an identifier of an object, since resolving a DOI results in metadata that identifies the object. This metadata may include one or more URLs where the object may be located, but may also include other information (described below).
- DOI is an actionable identifier
  - The purpose of the DOI System is to make the DOI an actionable identifier: a user can use a DOI to do something. The simplest action that a user can perform using a DOI is to locate the object that it identifies. Unlike a URL, however, a DOI can identify the object in greater detail since resolving a DOI results in metadata. Sometimes the DOI will necessarily resolve to metadata comprising the URL or network address of the object that it identifies. But, this is not always the case. Since the DOI is a reference to metadata, the DOI can be used to identify classes of

    intellectual property – abstract "works", physical "manifestations", performances – that cannot be directly accessed in a digital file. The DOI System distinguishes what the DOI identifies from what the DOI may resolve to. The technology used to manage the resolution of the DOI is the Handle System (*IETF RFCs 3650, 3651, 3652*).

- DOI is an interoperable identifier
    - A "legacy" identifier can form an integral part of a DOI identifier.
    - DOIs are implemented using The Handle System: a protocol for general-purpose distributed information systems designed to provide global name services for use in networks such as the Internet.
    - DOI uses for its optional object specification a DOI Data Model (a.k.a. metadata) and the indecs Data Dictionary. The indecs Data Dictionary includes a subset of the *ISO MPEG 21 Rights Data Dictionary, ISO/IEC 21000*-6. Together, the metadata and data dictionary provide to the DOI system semantic interoperability with existing metadata element sets.
- DOI is a digital identifier of objects, not merely an identifier of digital objects.
    - The indecs framework recognizes the concept of functional granularity. In other words, the indecs framework makes it possible to identify an object when there is a reason to distinguish it.. This is echoed in the DOI treatment of an identified object: through the use of extensible metadata, the identity of an object can be distinguished to the point necessary to distinguish it. The Handle system used by DOI uses TCP/IP but avoids the need to use the DNS.
- DOI supports multiple resolution
    - The Handle System is unlike most other resolution technologies in supporting multiple resolution. A DOI may have multiple data values of different types associated with it (email addresses and URLs, for example), and multiple data values of the same type (several URLs). The same DOI can resolve to different data, depending on the way in which the Handle System is queried.

<u>Part B. Logical components of a DOI system</u>

The DOI system has four logical components:

- Numbering (DOI syntax)
    - assigning an identifier to the object that the DOI identifies.
- Description (Metadata)
    - creating a description of the object that has been identified with DOI, through the DOI Data Model,. with storage of such metadata within the Handle system.
- Resolution
    - making the identifier "actionable" by providing information about to what the DOI should resolve
    - retrieving metadata from the Handle System to deliver the services that the actionable identifier can provide to users; and

    - operating on the metadata in such a way as to provide specific services to users.
- Policies:
    - the rules that govern the operation of the system.

Part C. Syntax of a DOI

The DOI system uses as its naming syntax the NISO standard *DOI syntax Z39.84*, which conforms to the functional requirements of the two generic approaches for naming entities on the Internet: the Uniform Resource Name (URN) and the Uniform Resource Identifier (URI).

The DOI has two components, known as the prefix and the suffix. These are separated by a forward slash. The two components together form the DOI:

10.1000/123456

In this example, the prefix is “10.1000” and the suffix is “123456” There is no technical limitation on the length of either the prefix or the suffix; in theory, at least, there is an infinite number of DOIs available.

The prefix itself has two components. All DOIs start with “10.” This distinguishes a DOI from any other implementation of the Handle System. The next element of the prefix is the alphanumeric string that is assigned to an organization that wishes to register DOIs. The prefix may be further divided into sub-prefixes, for example:

10.1000.10/123456

The DOI is an opaque string (a dumb number). No definitive information can or should be interpreted from the number in use. In particular, the fact that the DOI has a prefix issued by a particular organization should not be used to identify the owner of any given intellectual property – the DOI remains persistent through ownership changes, and the prefix is unaltered.

Following the prefix (separated by a forward slash) is a suffix that is unique to the prefix. The combination of a prefix for the Registrant and unique suffix provided by the Registrant avoids any necessity for the centralized allocation of DOI numbers. The unique suffix may be generated by a system used by the Registrant.

The DOI suffix can be any alphanumeric string that the Registrant chooses. This can simply be a sequential number, or it can make use of an existing (legacy) identifier. An example of a legacy identifier is an ISBN.

The issuing of unique prefixes to Registrant organizations places the onus on those organizations to ensure that the DOIs that they are registering are indeed unique.

The DOI is case insensitive and may incorporate any printable characters from the Universal Character Set (UCS-2), of ISO/IEC 10646, which is the character set defined by Unicode v2.0.

Part D. Resolution of a DOI

Objects identified by a DOI may be of any form, including abstractions (e.g. as identified by ISTC). Resolution is the process of submitting a DOI to a network service and receiving in return one or more pieces of current information related to the identified object.

The DOI uses the Handle System to implement resolution. The resolution of a DOI (e.g. 10.1000/140) results in one or more pieces of typed data ("associated values"). Examples of typed data are a URL, an e-mail addresses, another DOI, and metadata.

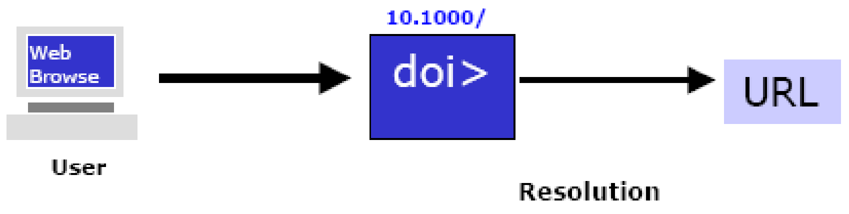


Figure 2 Example of DOI to URL single resolution

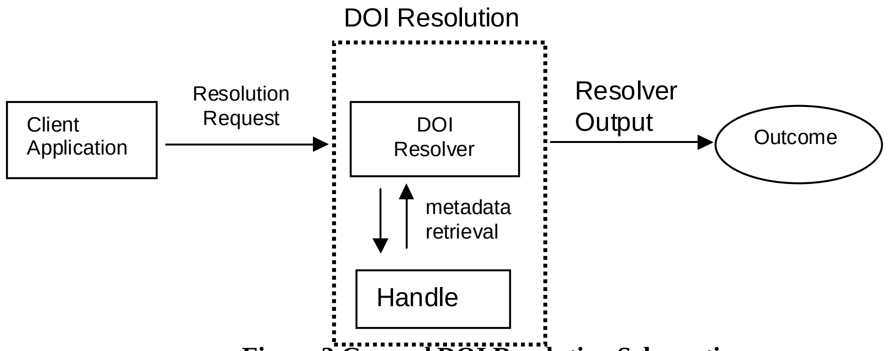


Figure 3 General DOI Resolution Schematic

Figure 3 is a schematic showing the data flow between the actors involved in DOI resolution. First, on behalf of a user a client application makes a resolution request to a DOI Resolver that performs a specific function and understands the interface with the client application. Second, the DOI Resolver retrieves metadata from the Handle system for the DOI that is the subject of the resolution request. Third, the DOI Resolver acts upon the DOI metadata and any additional metadata provided in the resolution request and produces an appropriate outcome.

Using multiple resolution, a DOI can be resolved to an arbitrary number of different associated values. Resolution requests may return all associated values of current information, individual values, or all values of one data type. These associated values can be displayed in a menu in a client application, from which the user may select the desired value. Moreover, a client application may deliver messages to a user wherein the messages are constructed from a syntax that allows the user to select the desired value. In addition these values may be processed automatically to achieve the objective of the resolution system.

Current Web browser technology requires additional functionality to allow the browser to make full use of DOIs: additional browser features are necessary. It is anticipated that features supporting resolution will commonly be built into browsers in the future.

There is a freely available “resolver plug in” that can be downloaded from http://www.handle.net/resolver/. For both Netscape and Microsoft IE browsers, the plug-in extends the browser's functionality so that it understands the Handle protocol.

Alternatively, without the need to extend the Web browsers' capability, DOIs may be structured to use the default public DOI proxy server (http://dx.doi.org). The resolution of the DOI in this case depends on the use of URL syntax. For example, “doi:10.123/456” would be written as http://dx.doi.org/10.123/456.

Part E. The Handle System

Background: The Handle System is a general-purpose distributed information system designed to provide an efficient, extensible, and secured global name service for use on networks such as the Internet. The Handle System includes an open set of protocols, a namespace, and a reference implementation of the protocols. The protocols enable a distributed computer system to store names, or handles, of digital resources and resolve those handles into the information necessary to locate, access, and otherwise make use of the resources. These associated values can be changed as needed to reflect the current state of the identified resource without changing the handle, thus allowing the name of the item to persist over changes of location and other current state information. Each handle may have its own administrator(s) and administration can be done in a distributed environment. The name-to-value bindings may also be secured, allowing handles to be used in trust management applications..

Within the DOI system, an implementation of the Handle System is used for DOI resolution. The Handle System is a protocol specification. An implementation of a Handle System is made up of local handle services (LHS). A local handle service is made up of one or more sites. A site is made up of one ore more handle servers. Handle servers store handles.

One local handle service is unique, the Global Handle Registry®: the handles it stores, which are the naming authority handles, makes it the LHS to query to find out which services store all the other handles.

Part F. DOI Data Model

Metadata may include names, identifiers, descriptions, types, classifications, locations, times, measurements, relationships and any other kind of information related to a Resource.

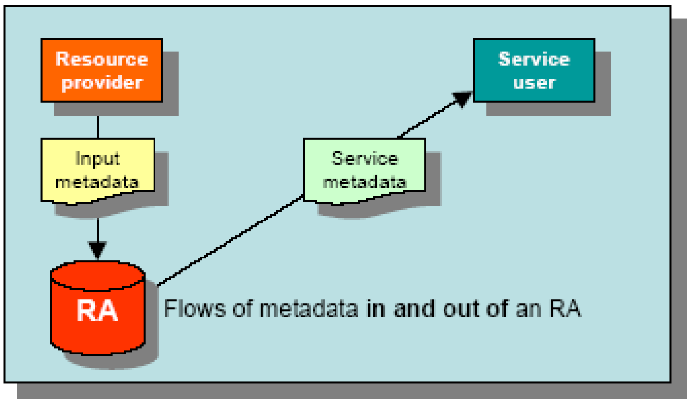


Figure 4 Flows of metadata in and out of an RA

There are two ways in which every DOI Registration Agency (RA) is bound to deal with metadata. An RA will gather input metadata from Resource providers; and an RA will need to provide some level of output or service metadata to support DOI services. Input metadata will provide some, but not necessarily all, of the service metadata. In some cases, a metadata declaration will itself be a complete DOI service (for example, "provide an ONIX Product message for this Resource"). These two flows of metadata declarations are illustrated in Figure 4.

DOI policy places no restrictions on the form and content of an RA's input and service metadata declarations, except insofar as input metadata must support the minimum requirements implicit in the DOI Kernel (see below). RAs may specify their own metadata schemes and messages, or use any existing schemes in whole or part for their input and service metadata declarations.

Part G. The responsibilities of a DOI Registration Agency

RAs are responsible for the administration an operation of the DOI system in the following areas:

- The RA must provide a service for the initial registration and subsequent maintenance of DOIs and their metadata into the Handle system.
- Should the RA choose to operate its own resolution service, such service must at a minimum perform the same functions as the public default resolution service at http://dx.doi.org

- Should the RA choose to operate its own Local Handle Server such operation must comply with IDF specified procedures including the right of the IDF to operate backup servers for the RA's content hosted on their Local Handle Server.
- An RA must support the distributed deployment of the public default resolution service as specified by the IDF.
- An RA must be capable of producing a Kernel Metadata Declaration (a.k.a. DOI Kernel, see below) for each DOI issued. Metadata exchanged between RAs supporting DOI services should be exchanged using an agreed DOI Resource Metadata Declaration (RMD, see below) for the Resource or Service type. Proprietary terms (data elements and values) used by RAs in Kernel and Resource Metadata Declarations should be registered in the IDF's data dictionary (iDD).

Part H: DOI Kernel

The DOI Kernel, which is formally specified in an XML schema, answers a number of basic questions about the identified Resource (see Figure 5).

Questions about the Resource	Kernel element(s)
What is the DOI being allocated?	DOI
Is it commonly referenced with another identifier (e.g. an ISBN)?	resourceIdentifier(s)
What is it usually called?	resourceName(s)
Who is principally responsible for its creation or publication? What role did they play?	principalAgent(s), agentRole(s)
Is it a *physical fixation, a digital fixation, a performance, or* an *abstract work*?	StructuralType
How is it perceived -- is it *audio, visual, audiovisual,* or *abstract*?	mode(s)
What particular kind of Resource is it? (e.g. an *audio file, scientific journal, musical composition, dataset, serial article, eBook, pdf etc)*	ResourceType

Figure 5 Kernel Elements

There may also be a few questions about the issuing of the DOI and Kernel itself (see Figure 6).

Questions about the Resource	Kernel element(s)
Which RA issued this DOI?	RegistrationAgency
When was this Kernel issued?	IssueDate
Which version is it?	IssueNumber

Figure 6 Administrative Kernel Elements

Values of some Kernel elements (names and identifiers) are simply alphanumeric strings. The other elements are drawn from sets of allowed values: for example, an agentRole might be “Publisher”, “Composer” or “Distributor”.

Two Kernel elements (StructuralType and mode) have a small, prescribed set of allowed values which all RAs must recognize. For the other elements and sub-elements, RAs may use their own choice of values, and add to them as and when required. These value sets must be registered in the data dictionary (iDD) for mapping purposes, so that any application using Kernel metadata from more than one source may be capable of presenting an integrated set of values to its users. The Kernel Declaration described here applies to resources in the form of Creations (items of intellectual property which represent the scope of early DOI implementation).

Part I. indecs Data Dictionary

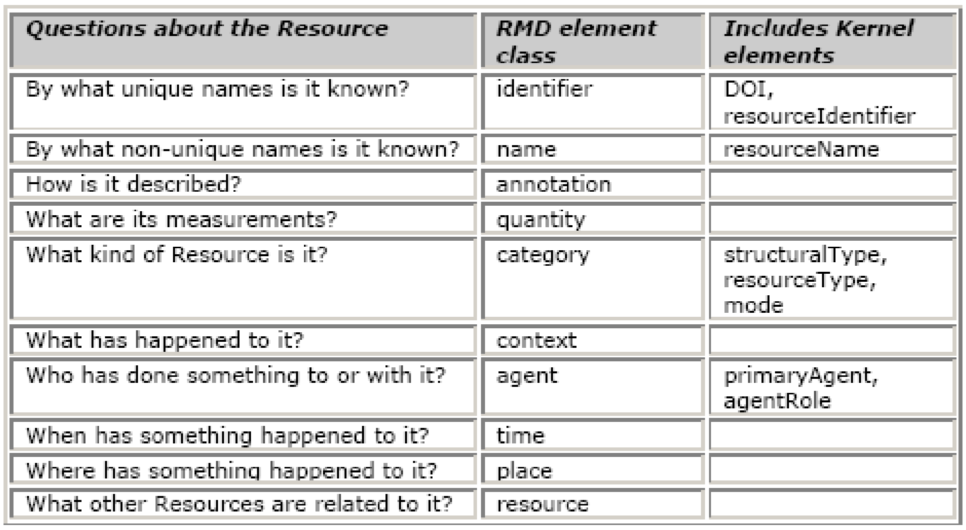
The indecs Data Dictionary (iDD) is the repository for all data elements and allowed values used in Kernel Metadata declarations and Resource Metadata Declarations (RMDs, see below).

The iDD enables the definition and ontology of all metadata elements to be available to all RAs, and provides the necessary mappings to support metadata integration and transformations required for data interchange between RAs.

iDD is a structured ontology compliant with logical axioms and constructors common to ontology languages such as W3C’s OWL (Web Ontology Language). It can, for example, support the production of legal OWL ontologies. All allowed values used by an RA in its Kernel Metadata, and all data elements used by an RA when mapping to an RMD, must be registered in the iDD.

Part J. DOI Resource Metadata Declaration

A DOI Resource Metadata Declaration (RMD) is a message designed specifically for metadata exchange between DOI RAs. An RMD is in the form of an XML document which conforms to an XML Schema (xsd). All its elements and allowed values are mapped into the iDD.



Questions about the Resource	*RMD element class*	*Includes Kernel elements*
By what unique names is it known?	identifier	DOI, resourceIdentifier
By what non-unique names is it known?	name	resourceName
How is it described?	annotation	
What are its measurements?	quantity	
What kind of Resource is it?	category	structuralType, resourceType, mode
What has happened to it?	context	
Who has done something to or with it?	agent	primaryAgent, agentRole
When has something happened to it?	time	
Where has something happened to it?	place	
What other Resources are related to it?	resource	

Figure 7 RMD basic element classes

The RMD uses a generic metadata structure of ten basic data element classes, developed from the indecs framework model. Figure 7 shows the ten RMD basic elements, and to which class each of the more specialized Kernel elements belong.

RMDs may incorporate data elements, allowed values, codes and composites from any other standard or proprietary message or metadata schemes (for example ONIX, SCORM or MARC) and draws on standard ISO codes and formats for Languages, Territories, Currencies, Measures and Dates and Times.

All element types and allowed values for an RMD are registered in the iDD. Every RA wishing to make use of an RMD must register the corresponding data elements and values in its own database to ensure reliable mapping by other RAs.

RAs are free, of course, to use existing standards to communicate metadata between them where they are suitable. If, for example, two RAs are providing services requiring ONIX metadata, then it would be expected for one to provide ONIX messages to the other.

An RMD is developed with contributions from two or more RAs. RMDs are available for use by any RA. Any RA making use of a specific RMD may contribute to the editorial development of the RMD. An RMD will include the metadata elements required for all nominated services by any participating RA. Specific data elements within an RMD may be required only for specific RAs or Application Profiles, enabling the same RMD to be used flexibly within a community.

Part K. DOI Application Profile

Each DOI is associated with one or more Application Profiles, and each AP is associated with one or more defined Services.

A Service is simply a defined result from a defined action, i.e., do X and the result will be Y. One of the services, possibly the only one for some DOIs, is the provision of kernel metadata for each DOI. Other sets of metadata may also be available for some DOIs and this, as with other services, would be known through the inclusion of a given DOI in an AP and the association of that AP with the given service.

Each AP is itself identified by a DOI, e.g. DOI:10.profile/7, and it is that DOI, instead of each of the content level DOIs, that is directly associated with the set of services relevant to each member of its set.

Each of the registered services is also identified by a DOI, e.g. 10.Service/3. Associated with each service is a natural language description of the service plus one or more interface descriptions (e.g., Interface Definition Language, Web Services Definition Language, or bindings and other information needed for using the service). The same service could be used by multiple APs through the simple mechanism of registering the service under multiple APs, just as a single AP will be 'used' by multiple DOIs.

APs may themselves be grouped into APs, i.e., they can be nested. This makes it possible to group them together for greater levels of indirection yet still maintain the flexibility to create new grou pings and associations as needed

Part L. DOI API

As procedures and structures are added for layering DOI information over the handle system it becomes increasingly important to define those procedures and structures in a separate layer. In other words, the added complexity of APs and Services requires the formal definition of a separate DOI layer.

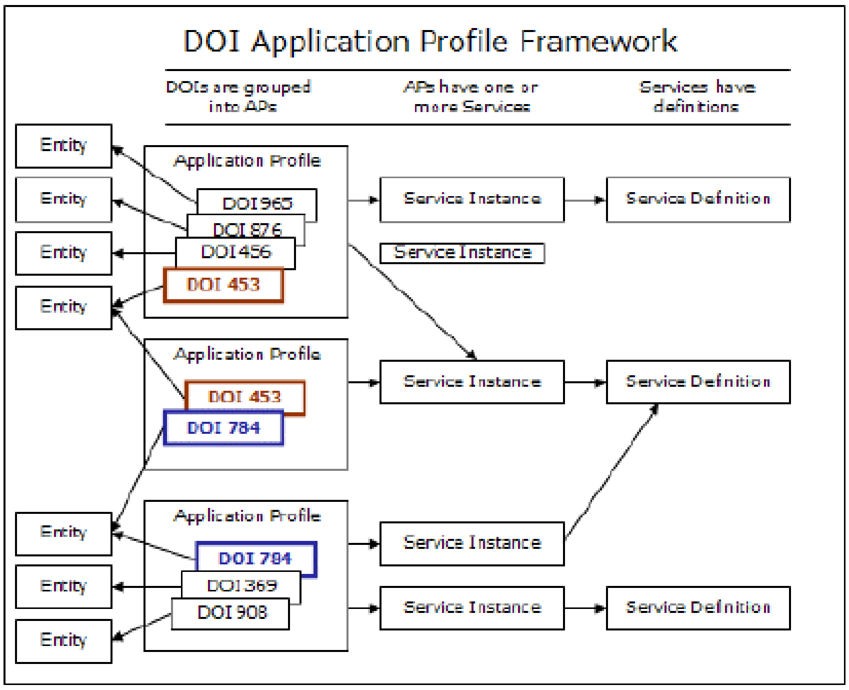


Figure 8 DOI Application Profile Framework

Figure 8 shows an abstract view of this model, which has also been referred to as the Application Profile Framework. DOIs are linked into Application Profiles. Any single DOI can be a member of multiple APs. Each AP can be linked into multiple Services. That linkage is to one or more specific instances of a Service. Each defined Service can be made available in multiple ways, referred to as instances. This makes it possible to add a Service to many DOIs by adding that Service to relatively few APs.

This framework is implemented in the handle system, using DOIs for both APs and Services and linking them together through typed handle values.

APIs (application programming interfaces) have been created that abstract out the details of the handle system implementation and make it possible to administer the structures and to use them in applications while ignoring the details of the implementation. They provide “hooks” down into the handle system without manipulating the handle records directly.

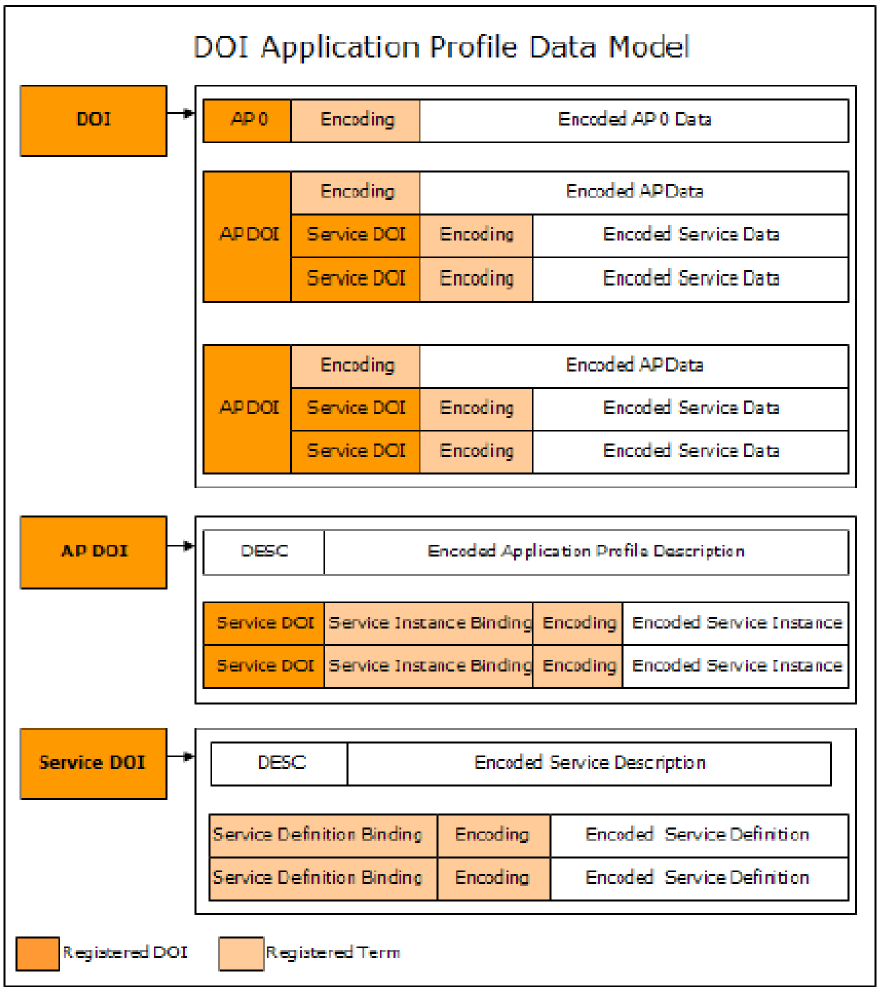


Figure 9 DOI Application Profile Data Model

Figure 9 shows the DOI data model from the API perspective. Here the details of the way in which DOIs, APs, and Services are connected is hidden from the application developer who is provided instead with methods for retrieving APs and Services associated with a given DOI.

Part M. Registration of DOIs and metadata

Registration Agencies support registration of DOIs with associated metadata declaration, i.e. using a DOI Application Profile. Individual Registration Agencies will develop their own workflow and procedures for the management of DOI registration, and metadata deposit and maintenance. The following procedures are representative:

> *Registering DOIs:* The following procedure is as an example of a process that could be used by an individual Registration Agency for the registration of a DOI with declared metadata.

This procedure allows for the batch registration of DOIs and associated metadata records into a DOI Central Metadata Directory run by the Registration Agency; this directory can subsequently be queried. The batch file format currently in use is XML as defined by a specific XML Schema, and submission is via HTTP POST. Security is HTTP basic authentication; PGP encryption will be added later. Batch receipt is confirmed to the sender via email.

Metadata Creation: The Registrant prepares XML batch files in accordance with the Schema; these are further constrained by a set of rules for the data, which define the expected content of each metadata element. An XML batch may contain metadata for hundreds of DOIs.

The development and implementation of quality control measures used to ensure the validity of the metadata content are the responsibility of the Registrant. Quality control and data checking can be assisted by processes put in place in the RA's metadata collection process.

Metadata Collection: The XML is validated upon receipt against the Schema. If the XML does not parse, the batch is refused; the Registrant must correct the XML and resubmit the batch.

XML batches are submitted to a named HTTP server via HTTP POST to a Java "servlet", which parses and validates the XML file, and notifies the Registrant in real time whether or not the XML is valid and has been accepted. The submission process captures and verifies a DOI System prefix holder login and password prior to validating the XML. The XML files themselves contain timestamps used as identifiers of the batch; should the Registrant so wish, each DOI record may have its own timestamp.

DOI Deposit: The servlet then deposits each DOI and its corresponding URL into the DOI System, adding timestamp data value. If the DOI is not new and therefore already exists in the DOI System, the timestamp is key to determining whether the DOI data being contributed is newer than the data that is already in the system; if so, the existing DOI data is updated. A log file also written in XML is created for each batch, indicating the total number of DOI records in the batch, the number of successful deposits into the DOI System, and the number of failures. For each failure, the DOI is provided, along with the reason for the failure. While DOI System failures may be the result of system errors, they are most typically caused by an attempt to overwrite existing DOI data with older data.

Metadata Database Record Generation: The original XML batch files, along with the log files for the batches, are made available daily to the metadata database deposit process, where they are indexed and then made available for searching. A final XML log file is generated to indicate the success of the database deposit (again, failures are due primarily to network or system errors) combined with those from the DOI deposit process, and this combined XML batch diagnostic is emailed to the Registrant.

The entire metadata collection process is expected to be completed and reported to the Registrant in as close to real time as possible; 24 hours is currently seen as a reasonable target time. However, when Registrants initially make deposits, there are large amounts of legacy material and coordination is needed on when the legacy batches are deposited or system performance can be affected.

Data Querying: The metadata database (MDDB) may be queried by submitting a batch file of known metadata fields in a specified format, currently pure ASCII text on separate lines, with fields delimited by vertical bars. The batch interface will query the database and return the corresponding DOIs (if known), or a diagnostic message. Batch query files are submitted by HTTP POST to a named HTTP server.

An RA has the authority to register a DOI and associated metadata to provide services that exist in repositories outside the DOI handle system. For example, an RA may register DOIs with URLs and some basic metadata in the DOI Handle system while keeping another set of metadata in its own repository.

Part N. DOIs & OpenURL

Background

The OpenURL Framework is a syntax for transporting metadata and/or identifiers about an object, using an established set of parameter names, to enable context-sensitive linking for the development of user-specific services. It is under development as a NISO standard, OpenURL Framework, Z39.88. The Proposed Standard, Registry, and KEV Implementation Guidelines are available from NISO Committee AX.

The OpenURL Framework includes DOI as one of its registered Namespaces and DOIs are widely used in OpenURL implementations. This documentation references only part of the OpenURL Framework Registry. More references to OpenURL and the DOI Proxy Server will be found in the documentation on Parameter Passing.

The DOI Proxy Server and OpenURL

In the OpenURL Format, descriptions of referenced resources, and descriptions of the associated resources that explain the context of the resource, are contained in ContextObjects that are transported using the HTTP protocol. ContextObjects use a

Key/Encoded-Value format to create a string of ampersand-delimited pairs. The values must be URL-encoded.

Of the five ContextObject Entities, one of them, the Referent, is required. Within the scholarly information community, the Referent will likely be a journal or journal article, a conference proceeding, or a book. The Identifier for the Referent is its DOI.

The DOI Proxy Server is a web server that understands the Handle System protocol. It is not an OpenURL Resolver per se, and does not provide services to an end-user that pertain to the Referent within the ContextObject of the OpenURL. When it receives an OpenURL, it finds the DOI in the string, resolves it, and re-directs the end-user's browser to that URL, ignoring all other ContextObject Entities.

The default syntax for a DOI resolution request to the DOI Proxy Server is:

http://dx.doi.org/10.1000/demo_DOI

The same DOI resolution request using OpenURL would be:

http://dx.doi.org/openurl?url_ver=Z39.88-2003&rft_id=doi:10.1000/demo_DOI

The OpenURL Format standard has been approved by the NISO voting members and is being readied for publication

Part O. Parameter Passing

Background
Before the DOI System and CrossRef came into existence, the scholarly publishing community implemented bilateral linking agreements that used parameters (name/value pairs) included in standard URLs to exchange data. This practice enabled them to gather information about requests coming to their sites, such as which other publisher's site a request came from, and from which journal and article. They could then implement special access rules, or establish pricing for their content based on who was requesting it.

At the time that the publishers began using DOIs, they also began thinking about how DOIs and the DOI Proxy Server could be used to facilitate the exchange of parameters, and remove the need for individual bilateral linking arrangements. A procedure, evolved over several years time, was agreed on by publishers who are now members of CrossRef, implemented in the DOI Proxy Server, and has come to be called 'Parameter Passing'.

The DOI Proxy Server and Parameter Passing
In Parameter Passing, there are two URLs involved, both of which may be query strings and/or include parameters: (1) the resolution request sent by the 'referrer' to http://dx.doi.org/ that has the DOI, and (2) the URL associated with that DOI, registered in the DOI System by the 'referent'. Parameter Passing requires joining the query strings

of those two URLs together to form an 'out-bound' link. The names of the parameters used in both strings must be unique and defined for all parties. The OpenURL Format was chosen for the URLs because it specifies a set of parameter names that can be used to eliminate the chance of naming conflicts. (See "Parameter Passing Via the DOI Proxy" for the OpenURL parameters applicable for use in Parameter Passing, and the specific Common CrossRef Parameter Set.)

The DOI Proxy Server accepts a resolution request in the form of an OpenURL. (See the separate document OpenURL for more information.) For example:

http://dx.doi.org/openurl?url_ver=z39.88-2003&rfr_id=ori:rid:crossref.org&rft_id=doi:10.1256/003590&rfr_dat=cr_setver%3d01%26cr_pub%3dSource%20Publisher%26cr_work%3dSource %20Journal%20Title%26cr_src%3dSRC-NAME

would be recognized by the proxy server as a Parameter Passing request. It will resolve the DOI, and then check the domain of the URL against an 'opt-in' list that identifies organizations participating in Parameter Passing.

If the URL is not in the opt-in list, the proxy server will redirect the user's browser to the registered URL. If the URL is in the opt-in list, the proxy server will construct a new URL as follows:

* Replace the registered URL's domain name and/or port number with different values, if replacements are specified in the opt-in list.

* Move all the parameters from the in-bound link to the out-bound link, except for the rft_dat parameter.

* For the rft_dat parameter, if the registered URL is an OpenURL, move the rft_dat parameter to the out-bound link. If it is not already in OpenURL format, hexencode the entire query string in the URL and place it into the out-bound link as the value of the rft_dat parameter.

The referent is assumed to have implemented a service capable of using the nested parameters. The assumption is that by agreeing to participate in Parameter Passing, a publisher will accept any and all parameters identified in the Common CrossRef Parameter Set.

Part P. Validation of DOIs

Objects identified using the DOI System are subject to verification. This may include an RA taking an action to determine whether the successful resolution of a DOI is possible. The determination reached by the action may be stored in the metadata associated with the DOI. A subsequent action by an RA may reveal that a previously resolvable DOI is no longer resolvable or vice versa. In response to this new information, a prior determination of the DOIs resolvable status may be deleted from the metadata associated with the DOI. Alternatively, the new information may be stored in metadata associated with the DOI.

Metadata used within the DOI System is also subject to verification. For example, during the update of metadata, an RA verifies the timestamp of the metadata to discern a time-forward update from a time-backward update.

RAs support the verification of a participant in the DOI System. This verification may comprise capturing and verifying the participant's credentials. The credentials may comprise a username and password.

The DOI System allows vendors of content in electronic form to control the material and restrict its usage in various ways that can be specified by the vendor.

Part Q. Caveats

This is a draft document.