TECHNICAL OVERVIEW

November 2010
# TABLE OF CONTENTS

1. **OVERVIEW**
   - 1.1 Document scope
   - 1.2 Background
   - 1.3 Technical principles
   - 1.4 Target users

2. **SYSTEM ARCHITECTURE**
   - 2.1 Registry requests
   - 2.2 Registry components
   - 2.3 ID Syntax

3. **EIDR DATA MODEL**
   - 3.1 Categorization of objects
   - 3.2 Sample applications to movie and TV assets
   - 3.3 Metadata Overview

4. **API AND APPLICATIONS**
   - 4.1 Application Programming Interfaces (API’s)
   - 4.2 Applications

5. **MANAGEMENT MODEL – PARTIES, USERS AND PERMISSIONS**

6. **KEY OPERATIONS POLICIES**
   - 6.1 Modification of records and ACL’s
   - 6.2 Permanence of records and Parties
   - 6.3 Accuracy and elimination of duplicates
   - 6.4 Failover

**APPENDIX A – EXAMPLES OF APPLICATION TO MOVIE AND TV OBJECTS**

**APPENDIX B – REFERENCES TO ADDITIONAL DOCUMENTS**
1. OVERVIEW

1.1 DOCUMENT SCOPE

This document provides a technical overview of the registry, including:

- Block-level system architecture
- The underlying data model for assets and relationships
- Example applications of that data model to movies and TV
- System management components, including parties, roles, and permissions
- Interface descriptions, including the public API, bulk ingestion, and the UI
- Summary of operating policies, including account management, data integrity, and fault tolerance

For more detailed technical information, please refer to the additional documentation listed in Appendix B.

1.2 BACKGROUND

Digital distribution of movies and television has resulted in an explosion in the number and variety of assets that need to be packaged, syndicated, distributed and tracked as they move from creation to consumption across an increasingly complex global media supply chain. Enabling and automating that supply chain requires a precise and universally understood way to identify assets and facilitate transactions in B2B workflows.

The Entertainment Identifier Registry (EIDR) was developed to provide the foundational namespace essential to making supply chain interactions more efficient, accurate, and automated. Its goal is to offer a low-cost mechanism for uniquely identifying the complete range of audiovisual assets and products relevant to B2B commerce, including abstract works, all varieties of commercial versions of assets, micro-assets such as clips, and specific instantiations such as encodings. It also offers the ability to describe a flexible array of relationships that may exist between assets, allowing assets to be identified in a hierarchy of related content. Obtaining and using an ID is intended to be cheap and easy. Use is designed to be open and unrestricted.

1.3 TECHNICAL PRINCIPLES

The guiding technical principles for EIDR are:

**Flexible Data Model:** There is high variance in automation practices in the content creation and distribution industry, and the number of underlying types of assets is large and growing. The EIDR data model is designed to allow identification of a wide variety of commercial assets, including: abstract works; specific embodiments such as language versions and edits; related material such as commentaries and promotions; groupings of objects, such as series and composites; pieces of an object that have independent
existence, such as clips and alternate audio tracks; and technical manifestations such as encodings. EIDR is designed for ease of integration into both new and existing workflows, and can be extended as new requirements, use cases, and objects emerge.

**Explicit Support for Relationships and Hierarchies:** Very few commercial video assets are standalone – they exist as clips of larger items, composites of smaller items, localized versions of an original, digital encodings of a particular edit of a film, and so on. The EIDR data model makes it easy to define and use relationships between objects. It allows users to create and traverse hierarchies of closely related objects, such as edits, encodings, language versions, and other variations. Likewise, it allows the creation of cross-hierarchical relationships between less tightly related items, such as the inclusion of a clip of a movie by a TV show.

**Centralized Registration:** A good ID is unique; an ID represents a single object, and a single object is represented by only one ID. In order to guarantee universal uniqueness, EIDR registrations go through a central system that uses a de-duplication module to guarantee that an object is unique. Once a unique ID is assigned to an object, the ID becomes a persistent and permanent part of the registry, available for use by the media and entertainment ecosystem.

**Decentralized Resolution:** Using an existing ID has to be efficient and reliable, and different users of an identifier will have different requirements. Although creation of IDs goes through the central EIDR Registry, lookups and queries can be done in a highly distributed way, with the accompanying benefits for scalability and availability. Further, EIDR member companies can request a local read-only copy of the database, either as a bulk request or as close to real-time mirroring.

**Use of Existing Standards and Infrastructure:** Any system for creating and using universal identifiers is complex, and a good way to keep costs down is to use proven pre-existing technology. EIDR is built on the DOI (Digital Object Identifier) standard, which provides the technical and social infrastructure for globally available persistent identifiers. DOI provides a core metadata definition, standard proxies for web-based resolution of identifiers, and mechanisms for translating between metadata domains. DOI uses the Handle System, built by the Corporation for National Research Initiatives (CNRI), for its underlying identifier resolution mechanism, which provides excellent failover, scalability, and security features. (See http://dorepository.org.) EIDR is built on open-source registry software developed by CNRI and uses other open source components as well (e.g., Apache Tomcat).

**Application Programming Interfaces:** An ID is not an end in itself; only by being used does it provide benefit. EIDR provides developers with a full set of REST APIs for integrating EIDR into existing applications and workflow and for creating new applications. The API provides access to registration of IDs, modification of underlying metadata, management of relationships, lookup and resolution of IDs, traversals of relationships and hierarchies, and fully general queries.

**Interoperability:** There are already dozens of ID formats for audiovisual assets – official standards (e.g. ISAN and ISRC), industry-led standards (such as Ad-ID), and proprietary numbering systems (such as in-house production tags) -- and it is unrealistic to expect them to disappear. On the other hand, it is highly desirable to know how an asset with one kind of ID relates to an asset with another kind of ID. EIDR
supports interoperability by allowing alternate IDs as first-class metadata. This allows linking other systems to EIDR, which also allows EIDR to function as an intermediary between other systems. For example, EIDR can be used for automated translations from retailer stock numbers to content owners' royalty tracking systems.

1.4 TARGET USERS

EIDR is intended for use as B2B service by all participants in the movie and television supply chain. This includes

- **Content owners**: TV and Movie producers and distributors, rights holders, syndicators
- **Post-production**: Companies that provide post-production services such as multi-platform delivery, media asset management, broadcast playout, etc.
- **Retailers**: Companies involved in selling, renting movie and entertainment related products either online or offline
- **Distributors**: Cable Networks, TV networks
- **Other Service Providers**: Vendors who develop and offer services using the registry APIs. These include services for VOD, enhanced metadata, anti-piracy, rights management solutions, reporting, and others.
- **Industry and trade associations**: Groups whose members stand to benefit from improved universal tracking of their assets – for example, rights management groups.

2. SYSTEM ARCHITECTURE

The registry is a set of custom modules built on CNRI’s Digital Object repository, which in turn is based on the DOI and Handle System resolution infrastructure. The DOI technical and social infrastructure guarantees that an EIDR ID is permanent and highly available. (See the Appendix for further information on the DOI Standard.)

2.1 REGISTRY REQUESTS

Three general classes of requests come into the registry:

**Create Object**: A request comes through the public API to create a new object, which is described by a set of metadata. If the metadata does not describe any existing object, an ID is allocated, and the metadata is stored in the repository and indexed for quick access.

**Modify Object**: A request comes with an EIDR ID and changes or additions to its metadata, including both descriptive information and relationship information. Once again, the registry checks that the changed object will not be a duplicate of any existing object before making the change.

**Retrieve Information**: There are several ways of getting information from the system. The first set – resolutions and traversals – takes an ID as input and returns information about that ID in a variety of
formats (resolutions) or information about other objects related to that ID (traversals.) The second set – queries – takes a set of constraints on metadata fields and returns the IDs of objects whose metadata records match the query.

All of these functions are covered by access control mechanisms.\(^1\)

### 2.2 REGISTRY COMPONENTS

This section gives a block diagram of the system architecture and describes how the components interact to fulfill these requests.

![Figure 1: EIDR System Architecture](image)

Each object in the registry is assigned a unique, universal, persistent identifier, or EIDR ID. The identifier is a Handle, composed of a prefix and a suffix. The prefix specifies the resolution system for the handle (in this case the EIDR Registry) and the suffix specifies an object within that system.

The EIDR system consists of the following modules:

\(^1\) Section 5 below provides more details on access control, requests, and other management functions of the registry.
Core Registry: This module is a customization and configuration of the CNRI Digital Object Repository. It performs various functions including registration, generation of unique identifiers, indexing, object storage management, and access control. Each object is assigned a unique ID upon registration.

EIDR Object Repository: This stores and provides access to registered objects; for EIDR, these objects are collections of metadata, not the media assets themselves. The metadata includes standard object information, relationships, and access control settings.

De-duplication: This module is called by the Core Registry to check for uniqueness of a newly created or modified object. The de-duplication module responds to a registry request with one of three outcomes.

  No Duplicate: The record submitted is unique.
  Duplicate: The record submitted is a duplicate of an existing object in the registry.
  Potential Duplicate: There is a high likelihood that the record submitted is a duplicate of a record or one of many records in the registry. Each request specifies whether a potential duplicate is rejected or sent for manual review.

Public API: A set of REST API calls that allows applications to make requests of the registry. Using these calls, services can make individual or batched calls, and request immediate failure or manual review of errors from the de-duplication system. The EIDR User Interface is built using this API.

Web User Interface: EIDR provides a web-based user interface primarily for search and lookup. The UI also supports the more common workflows for registration and modification.

Bulk Ingest: This application accepts up to 100,000 registration requests at a time and manages submitting them to the registry. It accepts both flat datasets and ones that have internal hierarchy (such as seasons, series, and episodes.) It provides an asynchronous status interface the results of which are compatible with direct requests to the registry.

Handle System: The DOI ecosystem is an application of the handle system, in the same way that http is an application built on top of TCP/IP. The handle system provides distributed lookup and resolution services.

Admin API: This API provides calls to manage accounts, users, and ACLs.

Admin Console: The console enables the registry operator to perform account and user management, access control, and other support functions. It is built using Admin APIs.

2.3 ID SYNTAX

The registry supports three kinds of ID, each with a prefix and suffix:

  - Asset ID: The prefix for an asset ID in the operational registry is 10.5240. Test versions of the registry can use other prefixes. The suffix of an asset ID is of the form XXXX-XXXX-XXXX-XXXX-XXXX-C, where X is a hexadecimal digit and C is the ISO 7064 Mod 37,36 check character. The check is computed as Mod37,36 rather than Mod17,16 to allow for future extension of the ID format.
The check character is computed only over the DOI suffix. It does not include the prefix because if
the prefix is wrong, it is highly probable that that the DOI will go to an incorrect resolution system
anyway. The registry separately sanity-checks the prefix of any DOI sent through its API.

Other constraints are:

- Interfaces must normalize the hex digits a-f to A-F, and use case-insensitive comparison for
  asset IDs.
- The check character is a required part of the ID, not an optional hint. An EIDR ID is not valid
  without it.
- The dashes in the number are for legibility, and are not part of the number. An interface
  that accepts EIDR IDs must accept the suffix with dashes as above, or with no dashes.
  Accepting other formats is not required, but it is not prohibited.

It is highly recommended that an AV Registry ID include the dashes when it is presented for human
consumption or intended for eventual human use.

- Party ID: This identifies a Party, such as a Registrant. The prefix in the operational registry is
  10.5237. Test versions of the registry can use other prefixes. The suffix is 2 – 64 characters long,
  composed of 0-9 (decimal digits), a-zA-Z (upper and lowercase letters), and the special characters
  _ (underscore), . (dot), # (number, hash, crunch, octothorp), ( (left paren), ) (right paren), and space.
- User ID: This identifies a user of the system. The prefix in the operational registry is 10.5238. Test
  versions of the registry can use other prefixes. The suffix is 2 – 32 characters long, from the same
  character set as Party ID, with the exception of the space character, which is not allowed.

See Section 5 for more details on parties and users.

3. EIDR DATA MODEL

This section describes the underlying concepts used to describe assets in the EIDR data model, some
example applications of these concepts to movie and television assets, and the outline structure of the data
model itself.

3.1 CATEGORIZATION OF OBJECTS

All objects in EIDR are categorized according to four basic concepts—type, inheritance, dependence and
relationship.

TYPE

'Type' is a much over-used word, and EIDR uses it in three ways:

- Object Type, which is equivalent to the meaning of 'type' in programming languages, and describes
  the data fields needed for a particular object
- Structural type, which relates to the level of abstractness of an object
- Referent Type, which relates to the nature of the underlying object

**Object Type:** This is an extension of the DOI Kernel metadata for an asset. See the appendix in the API Overview for details of mapping EIDR fields to DOI fields. When the unqualified word ‘type’ is used in EIDR documents, it refers to these Object Types.

There are several types of objects in the registry, which for convenience are divided into two classes:

**Basic Type:** This type covers the minimal possible object. It is sufficient for describing a wide variety of assets.

**Derived Types:** These types include all the information in the Basic Type, and add extra information for describing more complex objects. The Derived Types in EIDR are Edit, LanguageVariant, Series, Season, Episode, Composite, Clip, Encoding, and InteractiveMaterial.

The DOI specification provides and requires two other kinds of type – Structural Type and Referent Type. Both of these are represented as metadata fields in the Basic Type:

**Structural Type:** This must come from a set of four provided by DOI. They correspond to increasingly more specific manifestations of a work.²

<table>
<thead>
<tr>
<th>Structural Type</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>This is for objects that have no reality, such as a series container or the most basic concept of the original work.</td>
</tr>
<tr>
<td>Performance</td>
<td>This is for items that are a particular manifestation or version of something, such as the Director's Cut of a film or the Welsh-language version of a TV show.</td>
</tr>
<tr>
<td>Digital</td>
<td>This is a particular digital manifestation of a work, such as an MPEG-2 encoding of a movie.</td>
</tr>
<tr>
<td>Physical</td>
<td>This is for physical version of an object. EIDR will support this for physical films and tapes in a future release</td>
</tr>
</tbody>
</table>

**Referent Type:** In DOI terms, the referent is the item to which the DOI refers independent of any particular instantiation or meaning. The DOI handbook says "ReferentType typically describes the abstract nature of the content of a referent irrespective of its structuralType". For example, an object created as a movie is a movie whether it is being shown in a cinema, broadcast as an edited version over terrestrial TV, or streamed over the internet.

² ‘Abstract’ loosely corresponds to objects ‘in potentia’, as described by Aquinas and other medieval thinkers. ‘Performance’ roughly matches his objects ‘in actu.’ The first three might also loosely conform to Plotinus’ three hypostases, with the fourth corresponding to his notion of the physical world.
Referent Type | Use
---|---
Series | An abstraction that contains ordered or unordered individual items
Season | A second level of grouping below a Series
TV | Content that first appeared via broadcast
Movie | Content that first appeared in a theater (in the US) or a cinema (in most of the rest of the world.)
Short | Loosely defined, to cover miscellaneous content such as published outtakes, special segments, etc.
Web | Content that first appeared on the Web. This is different from content from elsewhere that has been made available on the web.
Interactive Material | Assets which are not strictly audio-visual. It covers DVD menus, interactive TV overlays, customized players, etc.
Composite | Any asset composed of multiple other assets that is not more precisely describable.

**INHERITANCE**

Most objects in the registry are related to each other as nodes in a tree. For example, all of the seasons and episodes of a series form a tree rooted in the series object. The registry also supports additional non-parental relationships, such as one object being included in a composite with items from outside its own hierarchy.

Items in a tree inherit certain fields from their parent. Only metadata from the Basic Type can be inherited. Furthermore, an object can only be part of one tree, so it has only a single chain of inheritance. See below under 'Relationships' for how objects interact with objects outside of their own ancestors and descendants.

All trees have a root object, which has no ancestors of its own. Items which have no descendants are leaves or leaf nodes, and items with ancestors and descendents are called internal nodes.

**DEPENDENCE**

An object may depend on another object in some way, by including a reference to it. There is no inheritance, and the metadata of dependents and objects on which they depend have only coincidental relationship to each other. For example, when encoding A refers to encoding B by reference, A is dependent on B, and when Composite C includes Clip K, C is dependent on K.

**RELATIONSHIPS**

A relationship is a casual term for the way in which two objects are connected. Relationships are described with one or more objects and some metadata. They are classified as:

**Lightweight relationships** - which have no inheritance; the objects to which they refer do not influence the underlying nature of the object on which the relationship exists. These relationships are used primarily
when moving around the object tree and connecting object trees to each other. The lightweight relationships are IsPackagingOf, IsPromotionFor, IsAdjunctTo, and IsAlternateContentFor.

**Dependence relationships** - where the objects to which the relationship refers have a strong bearing on the basic nature of the object on which the relationship exists. Importantly, this means that the objects referred to in the relationship need to be taken into account when checking for duplicates when an object is created or modified. The dependence relationships are IsCompositeOf and IsEncodingOf.

**Inheritance relationships** - where the object on which the relationship exists can inherit basic metadata fields from the object to which the relationship refers. With the exception is IsLanguageVariant, only one inheritance relationship may exist on an object. The Inheritance relationships are IsSeasonOf, IsEpisodeOf, IsEditOf, IsLanguageVariantOf, IsEncodingOf, and IsClipOf.

It will be obvious from the preceding that some combinations of object type, referent type, and relationship make sense, and some don't. The EIDR API Overview summarizes the legal combinations in the ‘Cheat Sheets’ section, and explains them in terms of API calls and data structures, which are more precise than narrative prose can be. The definitive statement of legal combinations is the table in REST Validation Rules (see Appendix B for both documents.)

### 3.2 SAMPLE APPLICATIONS TO MOVIE AND TV ASSETS

Here are some example combinations of Object Type, Structural Type, Referent Type, and relationships.

<table>
<thead>
<tr>
<th>Movie/TV Asset</th>
<th>Object Type</th>
<th>Structural Type</th>
<th>Referent Type</th>
<th>Relationship(s)</th>
<th>Uses and Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie A</td>
<td>Basic</td>
<td>Abstract</td>
<td>Movie</td>
<td>None</td>
<td>The most fundamental concept of a particular film. This is a root object.</td>
</tr>
<tr>
<td>Edit B</td>
<td>Edit</td>
<td>Performance</td>
<td>Movie</td>
<td>IsEditOf(A)</td>
<td>A particular version of movie A. This is an internal node in this example.</td>
</tr>
<tr>
<td>Language translation C</td>
<td>Language Variant</td>
<td>Performance</td>
<td>Movie</td>
<td>IsLanguageVariantOf(B)</td>
<td>The translated version of the edit. This is a leaf node.</td>
</tr>
<tr>
<td>TV series D</td>
<td>Series</td>
<td>Abstract</td>
<td>Series</td>
<td>None</td>
<td>The root of a series tree.</td>
</tr>
<tr>
<td>Season E of a TV series</td>
<td>Season</td>
<td>Abstract</td>
<td>Season</td>
<td>IsSeasonOf(D)</td>
<td>The second level of a tree based on Series D.</td>
</tr>
<tr>
<td>Episode F</td>
<td>Episode</td>
<td>Performance</td>
<td>TV</td>
<td>IsEpisodeOf(E)</td>
<td>Individual episode of Series D in Season E.</td>
</tr>
<tr>
<td>Language</td>
<td>Language</td>
<td>Performance</td>
<td>TV</td>
<td>IsLanguageVariant</td>
<td>Translated version of</td>
</tr>
</tbody>
</table>
### 3.3 Metadata Overview

Registering an object requires enough metadata to ensure that the object is unique. All metadata in EIDR serves one of two purposes: either it helps in the disambiguation of items, and aids in guaranteeing uniqueness, or it helps describe how objects are related to each other. This section is a high-level summary of the metadata for the Object Types and the relationships.

Additional information about all metadata fields and how to use them is available in other technical reference materials listed in Appendix B:

- EIDR Field Guide provides more detailed descriptions of each of the fields.
- EIDR API Overview provides guidance on how to use the API to combine types, values, and relationships when creating and modifying objects.
- EIDR REST API Validation Rules contains the rules that determine which combinations of types, values, and relationships are legal.
- EIDR Schema (eidr-base.xsd) contains the lists of controlled vocabulary.

### Publication Status of Objects

Objects in the registry are either Valid and visible to the general public or In-Development and visible only to the registrant and other parties explicitly authorized by the registrant. Most objects in the registry have a status of Valid. In-Development objects are for use in the limited cases where the existence of the registered work has to be kept secret for business reasons, e.g., pre-release works. Permission to view In-Development objects can be granted to other parties, for which see the section below on Access Control Lists.

An object is promoted to ‘Valid’ using the Promote() call when it no longer needs to be hidden.

### Object Types

**Basic Type:** The Basic Type has these important fields

<table>
<thead>
<tr>
<th>Movie/TV Asset</th>
<th>Object Type</th>
<th>Structural Type</th>
<th>Referent Type</th>
<th>Relationship(s)</th>
<th>Uses and Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation G (of episode)</td>
<td>Variant</td>
<td></td>
<td>(F)</td>
<td>the TV Episode F.</td>
<td></td>
</tr>
<tr>
<td>Webisode H</td>
<td>Episode</td>
<td>Performance</td>
<td>Web</td>
<td>IsEpisodeOf(D)</td>
<td>A web-only episode of Series D.</td>
</tr>
<tr>
<td>H.264 encoding I (of webisode)</td>
<td>Encoding</td>
<td>Digital</td>
<td>Web</td>
<td>IsEncodingOf(H)</td>
<td>A particular encoding of Webisode H.</td>
</tr>
</tbody>
</table>

See Appendix A for more detailed examples of the treatment of movie and TV assets in EIDR.
<table>
<thead>
<tr>
<th>Field</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural Type</td>
<td>See discussion above</td>
</tr>
<tr>
<td>Mode</td>
<td>Visual, audio-visual, or audio</td>
</tr>
<tr>
<td>Referent Type</td>
<td>See discussion above</td>
</tr>
<tr>
<td>Resource Name</td>
<td>Name of the object</td>
</tr>
<tr>
<td>Alternate Resource Name</td>
<td>Second name for the object; for example, a working title</td>
</tr>
<tr>
<td>Primary Language</td>
<td>The main language used in the asset. For example, for the original releases of the Star Trek movies it would be English, not Romulan or Klingon.</td>
</tr>
<tr>
<td>Secondary Language</td>
<td>Other languages available with this asset, for example Welsh subtitles or Swedish dubbing.</td>
</tr>
<tr>
<td>Principal Agent (or Producing Agent)</td>
<td>Used to identify the studio, producer, or other principle entity responsible for creation of the specific object</td>
</tr>
<tr>
<td>Release Date</td>
<td>This can be specified as a year or as year, month, and day. When only a year is known, the month and day should be left blank.</td>
</tr>
<tr>
<td>Country of Origin</td>
<td>The country of the PrincipalAgent of the work (if known); otherwise, the country from which the bulk of work actually came (if known) of the work; otherwise empty.</td>
</tr>
<tr>
<td>Status</td>
<td>Valid or In-Development</td>
</tr>
<tr>
<td>Approximate Length</td>
<td>The approximate length of the object</td>
</tr>
<tr>
<td>Alternate ID</td>
<td>One or more non-EIDR IDs also describing the object</td>
</tr>
<tr>
<td>Display Name</td>
<td>Alternate display name where desired. This is intended for use by applications.</td>
</tr>
<tr>
<td>Administrators</td>
<td>Lists the registrant, and optionally either or both of the entities believed to be the current asset holder and a backup contact</td>
</tr>
<tr>
<td>Credits</td>
<td>Optional director and up to two actors; included for purposes of distinguishing duplicates</td>
</tr>
<tr>
<td>Registrant Extra</td>
<td>Optional field for registrant-specific use</td>
</tr>
<tr>
<td>Description</td>
<td>String limited to 128 characters; this is not intended as a description of the underlying content, but of the specific object being registered; for example &quot;Brazilian Release, 25th Anniversary Director's Cut&quot; or &quot;PAL low-res streaming&quot;</td>
</tr>
</tbody>
</table>

**Other Types – General**: All the other object types (the derived types) include the metadata for the basic type. Some of the metadata from the basic type can be inherited from a parent object. See the REST Validation Guide and the Field Guide for details.
Edit: This is the catch-all for changes of content, and covers creative edits. It also includes format changes that are fundamental. It does not include format changes that are just the result of an encoding technology; for example NTSC to PAL is a new encoding (see below), but not a new edit.

Edits constitute an artistic change rather than an encoding change. These may include:

- New creative edits (e.g., director’s cut versus the theatrical cut); these can be described using the type’s ‘Class’ field
- Widescreen versus pan-and-scan
- 3D versus standard
- Colorized versus black and white

With rare exceptions (e.g., films in the tradition of *What’s Up Tiger Lily?*) a new language on its own is generally not a new version. Some edits may introduce a new language as part of the edit (e.g., edits to satisfy local censors). See below under ‘Languages’ for changing just the language of an object.

Every object of this type must have a parent, and so cannot be a root object.

Language Variant: This covers changes in the primary and secondary languages of the asset. An object of this type specifies whether its languages are an addition to any inherited languages or a replacement of them.

The API provides functions to return the fully evaluated list of languages for an object, taking all additions and replacements into account. There are also functions for returning language information that is inherited and language information defined on an object itself. See the Languages appendix in the API Overview for details.

Series, Seasons, and Episodes: The Series type is intended for registration and management of content where several pieces are grouped in some way. An object of this type can be ordered, in which case all of its direct children have a sequence number, and the sequence number of each child is the principal means of disambiguation. For an unordered series, the name of each child is the principal differentiator. An object of this type must be a root object.

The Season type is an optional level below a Series, for grouping of temporally related items. An object of this type can be ordered or unordered, with the same rules for its children as and must be the child of an object of type Series.

An object of type Episode can be the child of an object of type Season or an object of type Series. A Series object can have Seasons as direct children, or Episodes as direct children, but not both.

---

3 The Series type is not intended for franchise management, but for more fundamentally connected assets.
**Composite:** This type describes objects that are assembled from other objects or pieces of other objects. The description of each source piece can include:

- Which piece of a source object is used
- Where that piece is used
- Which components of that source (audio, video, etc) are used

A Composite also specifies a Class indicating the nature of the composite. Currently the system can only reference EIDR objects as source for an object of type Composite. A future version will extend this to cover objects described by alternate IDs as well.

**Clip:** The Clip type is used for objects that are pieces of other objects. At a minimum, it contains the object from which it is derived. An object of this type can also provide information on:

- How long the clip is
- Which components of the parent object are used (audio, video, etc)
- The time in the parent of the start of the clip

An object of this type must always be the child of another object (the one from which it is derived).

**Interactive Material:** This type is used for items that are not audio-visual, such as DVD menu frameworks, games, and components for interactivity. Metadata includes:

- Format – a technical category, such as Flash, CMX, BD-J, HTML5
- Use – such as Menu, Skins, Overlay Game
- Type – e.g., Text, Executable, Metadata

**Encodings:** An encoding describes a particular distributable instantiation of a work. The current version of EIDR supports digital encodings; future versions will support analog formats as well. An encoding is registered as a child object.

The metadata for an encoding object is structured as:

- Basic information, such as size, aggregate bitrate, and a Class
- Optional encoding blocks for zero or more subcomponents:
- Audio, describing the format of the source material and the encoded output
- Video, describing the format of the source material and the encoded output, as well as any subtitle carried in the video
- Subtitles (carried separately)
- InteractiveMaterial
- Wrapper, describing the final packaging of all the components.

Depending on the workflow, an encoding can be registered in a variety of ways:

- An encoding can be registered for individual components, such as a language track or the video. In this case, only the appropriate encoding block is provided.
• An encoding can be registered with just the wrapper encoding block, which in turn references the EIDR IDs of registered individual components.

• An encoding can be registered with encoding blocks for audio, video, wrapper, etc. This would be the case if the registrant wants to record information on the component encodings, but does not want or need an ID for them.

• An encoding can be registered using a combination of the above, for example a video encoding block and a wrapper that refers to two separately encoded and registered audio tracks.

The encoding information is not intended to be a complete description of an encoded object; it needs to be sufficient for disambiguation at registration time and useful for managing the objects in the post-production and distribution workflow.

Sub-elements in the Encoding type with types beginning md: are taken from the Common Metadata Specification.\(^4\) The controlled vocabulary described there is overly general for EIDR's purpose, but is included for interoperability and ease of implementation.

LIGHTWEIGHT RELATIONSHIPS

Many of the object types above define relationships between objects; for instance, an object of type Episode 'is an episode' of some other object. The relationships that are based on type are either Dependence relationships or Inheritance relationships (or both).

Lightweight relationships are all structured similarly and contain the ID of the object to which the relationship is made, and a Class. The types are:

• IsPromotionFor
• IsAdjunctTo
• IsAlternateContentFor
• IsPackagingOf

Unlike other relationships, which can only exist singly, multiple lightweight relationships can exist on the same object.

A WORD ON ALTERNATE ID’S

The Alternate ID field is of particular significance in the EIDR metadata schema. It plays an important role in ensuring the interoperability of EIDR IDs with other existing ID systems.

The field consists of a type and a value. For example, an Alternate ID could have a type of ISRC and a value of FR-UM0-99-12345. Proprietary IDs are supported as well, with an added attribute giving the domain within which the ID is valid.

As another concrete example, the field could include references to a work’s ISAN, allowing cross-referencing between ISAN-registered works and the full hierarchy of commercial edits, language variants, encodings, and other assets that may be registered in EIDR for the same family of assets. If the work’s ISAN is given as an alternate ID for an object at or near the root of an EIDR tree, it can be found from any of that object’s descendants.

The Alternate ID field can also be used by metadata vendors to link EIDR records to vendor IDs that reference external sources of commercial metadata for the asset. Studios or other content producers may cross-reference to internal IDs used for other distribution or tracking purposes. In the end, one of EIDR’s principal goals is to serve as a useful cross-referencing tool for access to a wide variety of external sources of data about each registered asset.

4. API AND APPLICATIONS

4.1 APPLICATION PROGRAMMING INTERFACES (API’S)

The EIDR Registry provides REST-based APIs for creating, modifying, and controlling access to records as well as interfaces for lookup and search. It also provides a set of Administrative API calls that are intended to be used for non-asset functions involved in maintaining the registry, such as creation of parties and users. A detailed description of both public and administrative APIs supported by EIDR can be found in the EIDR API Overview referenced in the Appendix.

For a detailed description of the EIDR API, please see the EIDR API Overview referenced in the Appendix.

OBJECT CREATION AND MODIFICATION

The following categories provide calls for the creation and modification of objects. All calls are subject to permission checks, for which see Section 5.

Create: This is used to create an object of a particular type. To be successfully created, an object must pass the validation rules for legal combinations types, values, and relationships. If it is created as Valid, it must not be a duplicate of any other object in the system. In-Development objects are checked for being duplicates of existing objects when they are promoted, rather than when they are created.

Add, Remove, and Replace Relationships: Allow caller to create or delete relationships between two objects. All of these are subject to the detailed validation rules.

Modify: Modifying objects is done by getting current information on the object, modifying it, and then submitting the modification, which must pass validation and de-duplication.

Promote: This call turns an InDev object into a Valid object. Once it is made, the referenced object is visible to the world at large. InDev objects do not go through the de-duplication system when they are created, but are checked for duplicates when promoted.
Delete: Ideally, every EIDR ID is permanent. However, sometimes an item really does need to be deleted (for example, a cancelled project at a defunct production house), or is so badly wrong it should be removed. The registry does not delete the EIDR ID itself; rather, it makes the ID an alias (see below) for a standard tombstone object.

Alias: An alias is a simple indirection from one ID to another. An alias is not intended as a general tool; rather, it is intended to be used for correcting errors.

Request Status: All asynchronous calls return a token which can be used to query the status of the request.

OBJECT INFORMATION
There are three ways of getting information about an EIDR object using the APIs.

Resolution: provides various views of the metadata associated with a particular ID. Metadata can be requested in these formats:

- **Full**: this gives an object's complete metadata, filling in inherited fields and fully realizing any language additions or replacements.
- **DOIKernel**: returns metadata formatted according to the DOI Kernel metadata for a Creation.
- **SelfDefined**: returns metadata found on the object itself, ignoring inheritance and language accumulation.
- **Inherited**: returns only metadata which has been inherited from an ancestor
- **Simple**: returns minimal information, including the name, structural type, referent type, primary language, release date, and status of an object, along with skeletal descriptions of its relationships.
- **Provenance**: returns information about an object's creation and modification history.

Traversals: provide information about how the ID fits into the object hierarchy. Traversals include calls to find ancestors and descendants, which can be filtered by various criteria; calls for finding remotest relatives; and functions that return specialized information relating to the components of series.

Queries return the IDs of objects with metadata matching certain criteria. The queries can be done over the entire registry database or over an object and its descendents. The query language supports comparisons an various types of fields (dates, durations, strings, controlled vocabulary, languages) and a set of logical operators (AND, OR, NOT.) Fields to be tested are described with XPath syntax.

**MODES - ASYNCHRONOUS AND IMMEDIATE**

Calls can be immediate, in which case the result is returned immediately; or asynchronous, in which case the call returns a token which is used to discover the status of the request. Not all calls support asynchronous results; those that do have a flag in the interface specifying which mode to use.

There is a GetStatus(token) call for checking on the progress of asynchronous calls.
MODES – BATCH AND SINGLE

Some API calls can be submitted in a batch. A batch can contain only calls of the same type—e.g., a set of Create() calls. The order of execution of the calls within the batch is undetermined, so a batch should not include calls that depend on the results of another call within the batch. A token is generated for the batch itself; using the batch token, a token is acquired for each request within the batch.

4.2 APPLICATIONS

BULK INGESTION

Many prospective registrants have large back catalogs that could be registered with EIDR. Some of the items in that set of assets depend on other items in the set—for example, an episode depends on its parent series, and a translation depends on its original film. EIDR provides a mechanism for registering bulk content, consisting of:

- a schema for data preparation in a standard format
- registration and serialization (starting with items that have no dependencies)
- iterative results processing that feeds information back into the serialization process
- results reporting

For a detailed account of the EIDR bulk ingestion mechanism, please see the EIDR Bulk Registration Guidelines referenced in the Appendix.

USER INTERFACE

The EIDR user interface supports common workflows for registration and modification of objects as well as search and lookup. The UI is built on the EIDR API and provides all operations supported by the registry. The following are some of use cases supported by the UI.

- Resolve an ID – View its metadata and relationships
- Search for records based on one or more known object attributes (i.e., metadata fields)
- Create objects, with special screens for common cases (such as TV series)
- Modify objects
- Add or remove relationships for a selected object
- Create objects similar to an existing object in the registry
- Check status of submitted tasks
5. MANAGEMENT MODEL – PARTIES, USERS AND PERMISSIONS

This section describes the basic concepts and interfaces for interaction between users and the registry.

A party represents an entity such as a registrant or a producing agent. A user is an individual and is associated with a party. (A user can also be an abstract thing that can be treated as an individual.) All users are associated with a party.

All requests to the registry contain authentication information for a party and a user.

The user requesting the creation of a new object must be associated with the party identified as the registrant in the registration data. Only parties have permissions in the system; a user has all the permissions associated with its parent party.

PARTIES AND USERS

There is a predefined party representing the EIDR Registration Authority. Other parties can be one or more of the following:

Producing Agent/Principal Agent: These bring forth the object being registered: a studio in the case of an abstract work; or an encoding house for the work in a final digital form; or even an anti-piracy vendor for registering a new illegal copy of a work. The important thing to remember about this field is that it refers to the entity that most recently ‘touched’ the item.
Registrants, who register items: A registrant may be a Producing Agent, such as a studio or an encoding house, or it may be an external entity such as one doing bulk registration of back-catalogue items.

Current Asset Holder: This party may be included optionally by the registrant to indicate the entity the registrant thinks is most likely to have some authority over this object. For example, a metadata provider doing bulk registration may make this the same as the Producing Agent for objects reasonably sure to be under the control of that entity, and leave it blank otherwise.

Backup Contact: Used optionally to identify another entity that may be able to sort out things that are otherwise beyond the ken of the registrant.

Encoding Agent: Optional entity used when registering encodings.

Additionally, a party can be registered as:

Writer: Can read and modify objects, but not create them.

Reader: If on an object's Access Control List (ACL) can read In-Development objects that would otherwise be hidden. This matters only for In-Development (InDev) objects.

As an example, an encoding house that registers new encodings is the registrant for the object. In the strictest sense, they could be the Producing Agent as well, but that is not mandated; for example, the rights holder could require that the Producing Agent for the encoding be the same as that for the parent object. The encoding house is also probably the Encoding Agent for each item, unless some aspect has been subcontracted out elsewhere (e.g., to a specialist subtitle shop.)

A party can be either Active or Inactive. An Inactive Party may not make any modifications to the database; that is, it may not be a registrant or a Writer, and all users associated with it are similarly restricted.

Only the EIDR Registration Authority can create or modify parties. Users may request that a new party be added and request the roles the party can take on. For example, a registrant might want to add “Acme Studios” as a party for use as a ProducingAgent.

PERMISSIONS, ALLOWED ROLES, AND ACCESS CONTROL LISTS (ACL)

A regular record can be created, modified, aliased, or deleted. An In-Development record can also be promoted. It is also possible to read an object's administrative information, called its Provenance.

A party (and its users) can only create a new record if it has 'Registrant' in the AllowedRoles field.

A party (and its users) must be able to read an object in order to create or modify any inheritance, dependence, or lightweight relationships involving it. For example, an IsAdjunctTo relationship cannot be made to an InDev object by a party that is not on the InDev object's ReadACL.

Other actions are gated by ACLs on the individual records. Each regular record has an ACL for:

Modify: can contain entities that are of type Registrant or Writer. Required to modify an object.
Delete: can contain entities that are of type Registrant or Writer. Required to alias or delete an object.

ReadACL: can contain entities that are of type Registrant, Writer, or Reader. Required to read any ACL.

WriteACL: can contain entities that are of type Registrant. Required to modify any ACL.

ReadProvenance: can contain entities that are of type Registrant, Writer, or Reader. Required to read the provenance metadata. Only the EIDR Registration Authority can modify a record’s Provenance information.

In-Development records have two more ACLs:

Promote: entities on this list must be of type Registrant. Only entities on this list can promote an InDev object to Valid.

View: entities on this list can be of any type. Only entities on this list can view an InDev object or have them returned from a query.

Further details on the EIDR management API’s are contained in the EIDR API Reference document.

6. KEY OPERATIONS POLICIES

The following section addresses a few key user and operations policies most relevant to the this technical overview. General customer support information is available from the operations team and in other user manuals.

6.1 MODIFICATION OF RECORDS AND ACL’S

By policy, permission to modify a record in the EIDR Registry is initially restricted to the registrant. The registrant can add other parties to the ACLs. Any other party can request write access to a particular record. The request first is routed to the registrant who is encouraged to grant access to the Current Asset Holder as a matter of EIDR policy and to other parties as long as the requestor has a valid reason to modify the record and has appropriate EIDR credentials for writing to the registry. If there is a dispute relating to the ACL that is not resolved by the parties, the EIDR Registration Authority steps in to adjudicate according to EIDR policy guidelines.

Although EIDR does not track ownership of assets, it is the policy of EIDR that the Current Asset Holder when known is added to the ACL of a record upon request. It is also possible that in some circumstances a registrant or other party might be removed from the ACL in order to resolve a dispute or in response to activity that violates EIDR terms of use.

6.2 PERMANENCE OF RECORDS AND PARTIES

Once created, registrants are permanent in the registry and records created by them list them as the registrant. However, if a party terminates its EIDR Registration membership, its write access to the registry is discontinued according to its membership status.
The same is true for all parties created in the registry. Once created, Registrants, Producing Agents, Encoding Agents, Current Asset Holders and other Parties are permanent, and only the EIDR Registration Authority may modify Parties or create new ones.

Records also are intended to be permanent. The EIDR terms of use contain restrictions limiting the ability to delete a record. All records should be permanent and persistent absent special circumstances allowing aliasing or other extraordinary changes to the registry. If a record is inaccurate or otherwise corrupted and really does need to be deleted, the EIDR Registration Authority does not delete the ID itself, but instead makes the ID an alias for a standard tombstone object.

6.3 ACCURACY AND ELIMINATION OF DUPLICATES

All registrants with write privileges have to meet certain minimum eligibility criteria set by the EIDR organization and agree to the EIDR terms of use governing accuracy of data submitted to the registry.

Every record submitted to EIDR is run through a sophisticated system which uses an automated matching algorithm to determine if it is a duplicate of an existing record in the registry. While most submissions can be identified as unique or duplicate with high probability using the automated system, some require manual review to make that determination. This task is handled by the EIDR Operations staff.

Any user of the registry may report errors in records. The errors could be in the metadata describing the asset or in the administrative data related to the asset. The EIDR operations team is responsible for reviewing the report to determine if indeed there is an error in the record and then taking appropriate action.

EIDR places strong emphasis on data accuracy and takes steps to actively monitor quality and promote best practices among registrants.

6.4 FAILOVER

The registry operates with redundant systems built to be reliable over long periods of operation. Infrastructure from the DOI and Handle systems allows multiple systems to provide resolution services, which in turn provides highly available reads and queries.

The registry provides a mirroring API that can be used to request a full export of the registry, export of all items modified since a particular time, or repeating updates at a specified time interval. Members are allowed and encouraged to maintain local copies of the database to use for reference and backup purposes.
7. APPENDIX A – EXAMPLES OF APPLICATION TO MOVIE AND TV OBJECTS

**TV SERIES**

- **TV Series**
  - **Season 3**
  - **Season 2**
  - **Season 1**

**Abstract**

- **IsSeasonOf**
  - **Episode 3**
  - **Episode 2**
  - **Episode 1**

**Performance**

- **IsEpisodeOf**
  - **Promo**
  - **Episode 3**
  - **Episode 2**
  - **Episode 1**

**Digital**

- **IsClipOf**
  - **Clip**
  - **IsEncodingOf**
  - **iTunes encoding**
  - **Amazon encoding**
  - **IsLanguageOf**
  - **English**
  - **Spanish**
MOVIE - SIMPLE ENCODINGS AND RELATIONSHIPS

Abstract

Director's Cut (EN)

IsEditOf

Original Release (EN)

IsLanguageOf

French

IsClipOf

Clip

Performance

Movie

IsEditOf

Spanish

IsLanguageOf

Trailer (EN)

IsClipOf

IsLanguageOf

IsPromotionOf

Mezzanine

IsEncodingOf

iTunes

IsEncodingOf

Amazon

IsEncodingOf

DailyMotion

IsEncodingOf

YouTube

IsEncodingOf

iTunes.es

IsEncodingOf

Amazon.es
MOVIE - COMPLEX ENCODING

Abstract

Performance

Movie

Director’s Cut (EN)

Director’s Interview

Other Movie

Trailer

IsPackagingOf

IsCompositeOf

IsEncodingOf

IsLanguageOf

Movie Encoding

Interview Encoding

Interview Encoding

Movie Encoding

Spanish (audio only)

Spanish

English (subtitles)

Audio commentary

Component

Component

DVD

IsCompositeOf

IsCompositeOf

IsEncodingOf

IsEncodingOf

IsEncodingOf

IsClipOf

IsPromotionFor

IsAlternateContentFor

IsEditOf

IsAdjunctTo

IsPackagingOf
8. APPENDIX B – REFERENCES TO ADDITIONAL DOCUMENTS

For developers:

1. **EIDR API Overview** – a detailed walk-through of the calls and functionality of the EIDR REST-based APIs that enable users to perform asset operations such as creating, modifying and controlling access to records as well as lookup and search. It also describes Administrative API calls that are intended to be used for non-asset functions involved in maintaining the registry, such as, creation of parties and users etc.

2. **EIDR Field Guide** – a detailed walk-through of the EIDR metadata fields and their use.

3. **EIDR REST API Validation Rules** – describes legal combinations of types, values, and relationships for objects in the registry.

4. **EIDR REST Public API Reference** – Programmer's guide with syntax, return values, etc for the public REST interface.

5. **EIDR REST Admin API Reference** – Programmer's guide with syntax, return values, etc for the admin REST interface.

6. **EIDR Schema** – eidr-base.xsd is the XML schema for the EIDR Registry. It includes lists of controlled vocabulary and other useful information.

7. **EIDR Bulk Registration Guidelines** – description of EIDR policies and procedures for ingesting large numbers of records through bulk registration.

For general audiences

8. **EIDR Overview** – an informational slide set summarizing the EIDR Registry approach and organization.

9. **EIDR White Paper** - Universal Unique Identifiers in Movie and Television Supply Chain Management, describing the overall approach to development and operation of the EIDR Registry.

External specifications


11. **DOI References** –
