EIDR
Entertainment Identifier Registry

Registry Technical Overview

Table of Contents

1 Prerequisites

This document is primarily intended for developers who are undertaking an EIDR system integration directly via the EIDR HTTP API, using one of the provided EIDR SDKs, or by leveraging the EIDR Command-Line Tools. Other EIDR users with a technical background may also find it helpful in orienting them to the EIDR registry.

EIDR recommends that you also read the following documents, which contain more detailed information about the topics covered in this guide:

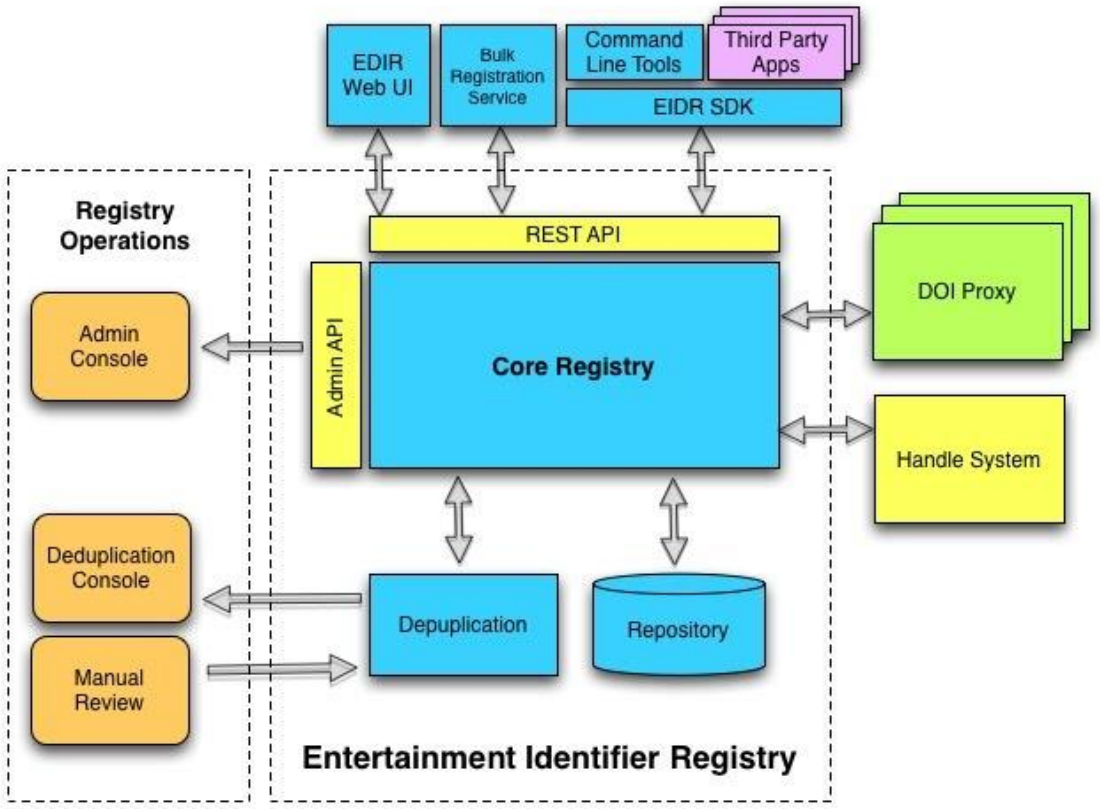
- ***Introduction to the EIDR Data Model***
- ***Programmers Guide***
- ***Best Practices and Use Cases for Abstraction Records***
- ***Data Fields Reference***
- ***REST API Reference***
- ***Schema Documents***
- ***EIDR: ID Format***

See https://members.eidr.org/ for additional documentation.

2 Overview

This document describes the usage of the EIDR Registry for technical users and developers, including how to read, create, and modify Content ID records, how to perform operations on Parties, and how to perform operations on Video Service identifiers. In addition, reference information for the HTTP API validation rules is included. For information on the Digital Object Identifier (DOI) Proxy, please see ***EIDR and the DOI Proxy***.

This diagram illustrates the EIDR system architecture:



Each object in the Registry is assigned a unique, universal, persistent identifier, or EIDR ID. The identifier is a Digital Object Identifier (DOI), a type of Handle conforming to ISO 26324 (for example, 10.5240/0000-0000-0000-0000-0000-X). The EIDR ID is composed of a prefix, which indicates the resolution system (in this case 10.5240 for the EIDR Content ID Registry), and a suffix (in this case 0000-0000-0000-0000-0000-X) identifying an object within that system, separated by a slash (/). For more information, see ***EIDR: ID Format***.

The EIDR Registry principally identifies commercial audiovisual content. It also identifies important corporate entities in the ecosystem, such as production companies (Parties) and distribution channels (Video Services).

The EIDR system consists of the following modules:

Core Registry: This module is a customization and configuration of the Corporation for National Research Initiatives (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.

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. For more information, see "De-Duplication" below.

REST API: An HTTP interface that provides access to the full set of non-administrative registry features. Using these calls, third-party services can perform all registry functions, including ID resolution, query, registration, and modification. Java and .NET SDKs built upon the REST API are available for application and service developers. The SDKs are available at https://members.eidr.org/ (a password is required to access them).

Command-Line Tools: The command-line tools are fairly simple applications, built on the SDKs, each of which provides a single function. They act as fully-commented source code examples and useful programs in their own right. A full set of command-line tools comes bundled with each SDK.

Web User Interface: EIDR provides a Web-based user interface primarily for metadata-based search and ID-based resolution. The UI also supports the more common workflows for registration and modification. The Web UI is built with the REST API. See https://ui.eidr.org/.

Bulk Registration Service: This application accepts any number of match, register, or modify requests and manages submitting them to the registry. It accepts both flat datasets and ones that have internal hierarchies (such as seasons, series, and episodes). This is one of several mechanisms for registering large catalogs after matching has been completed. For more information, see "Catalog Matching and Registration" below.

Handle System: The DOI ecosystem is an application of the Handle system, in the same way that HTTP is built on top of TCP/IP. The Handle system provides distributed lookup and resolution services. The worldwide DOI system is implemented using the Handle system from Corporation for National Research Initiatives (CNRI).

Usage Options for the EIDR Registry

You can access the EIDR Registry through its user interfaces, SDKs, and applications such as the bulk registration service and command-line tools. The following sections briefly describe these various forms of support.

There are two instances of the EIDR Registry: Production and Sandbox. The Production system is the official site, where all officially registered EIDR IDs exist.

The Sandbox is a development and test site where new users can start their practice registration efforts. Its data will generally not match the production data exactly, and the EIDR IDs do not necessarily match those in the Production system.

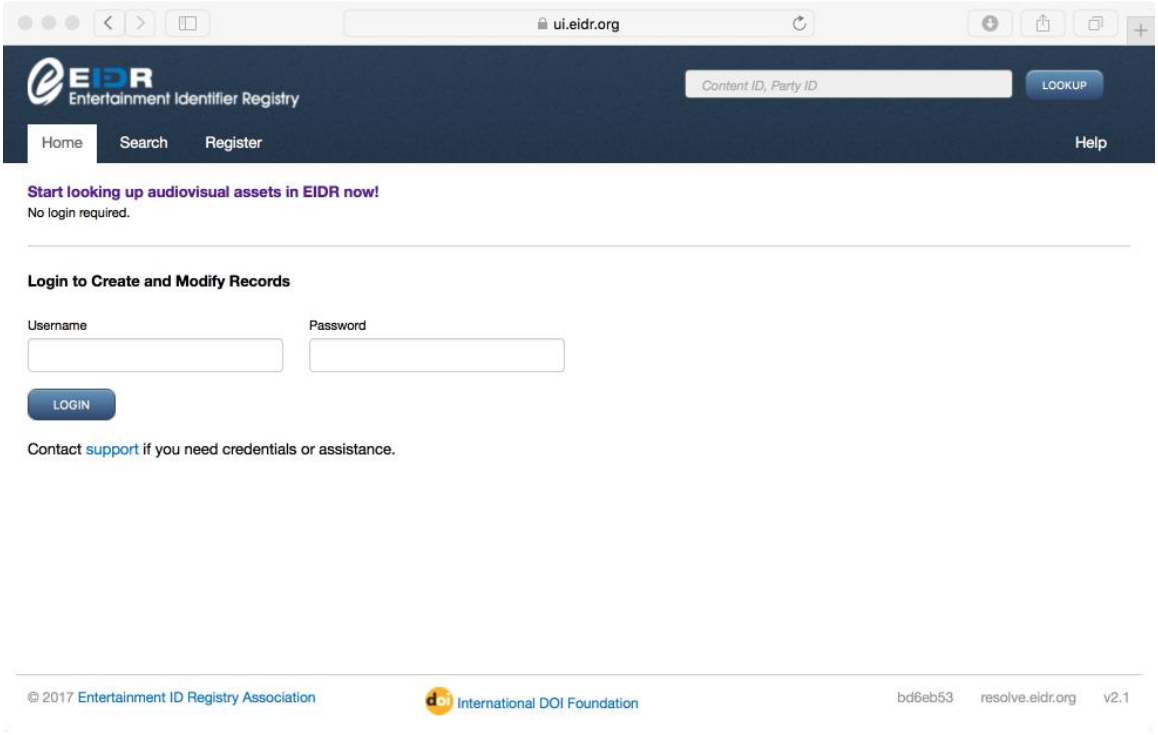
Do not assume that the records from the Sandbox are correct, as some of its data are transient. In addition, the Sandbox system is periodically wiped clean and replaced with a copy of the production system as a new starting point, deleting all practice efforts and corresponding EIDR IDs.

The EIDR Production Registry Web UI is available at https:// ui.eidr.org.

The EIDR Sandbox Web UI is available at https://sandboxui.eidr.org.

User Interfaces

The user interface is identical between the Production and Sandbox systems, so it is important that you use the correct one for your work. The Production masthead (below) is blue, while the Sandbox masthead is green.



In addition, you will find the linked registry name and version number in the lower right corner of the screen. This will generally be one of:

- registry1.eidr.org – The EIDR production primary (read/write) registry
- resolve.eidr.org – The EIDR read-only mirror registry
- sandbox1.eidr.org – The EIDR Sandbox registry

Through the user interface you can:

- Perform metadata-based searches
- Perform ID-based resolutions
- Look up the status of asynchronous requests using tokens returned by the requests
    - Initiated by specific users
    - Initiated by registrants
- Create new records
- Modify existing records
- Access the EIDR Support network

Command-line Tools

With the command-line tools you can perform every public registry function, including:

- Resolve EIDR IDs
- Examine the object hierarchy
- Make queries about Content records, Parties, and Video Services

- Create and modify objects (singly and in bulk)
- Manage relationships
- Promote objects
- Delete objects
- Alias objects

SDK

With the SDK you can create Java and .NET applications that make use of Registry services and perform Registry operations. If necessary, you can use the HTTP API directly, but this is not recommended.

3 Catalog Matching and Registration

You can match and register back catalogs with EIDR in several ways, but all of them have some common steps consisting of:

- Preparing data in a standard format, either XML or a spreadsheet
- An iterative process of matching the proposed records against the current EIDR database
- For records that were not found by the matching process (*gap* records)
    - Serialization: some records (e.g. episodes) may require that other records be registered first (such as registering a series before an episode.)
    - Registration of new content records
- Making corrections to existing records based on the matching results.
- Providing the matched and newly registered EIDR IDs to the EIDR user for inclusion in the EIDR user's metadata systems.

Most bulk registrations will have to go through the matching process. The matching process reduces the chance of de-duplication errors caused by the variability in quality of source material across providers.

To prepare large registrations:

1. Determine hierarchy compatibility. This includes determining the types of records you will be registering and how these fit with the types of records in your system. For example, if you are registering abstract works, determine which records represent the work in its general form. If you are registering digital assets (Manifestations), determine those versions (EIDR Edits) from which your asset originated.
2. Map your data fields to EIDR's and validate metadata compatibility.
3. Evaluate metadata for covering required EIDR fields and recommended EIDR practices (for example, Directors/Actors and Alternate IDs). Develop a plan to fill in any missing data.
4. Evaluate the quality and consistency of use of the resulting metadata (for example, release year, first billed vs. any four actors).
5. Match common production/distribution companies to those in EIDR's Party database.
6. Run the batch through match before registration. That way, you'll know if any records require further preparation and how many of them will require manual review. (Our standard SLA for registry response does not apply to catalog projects, so you'll must coordinate with EIDR Operations in advance to determine the SLA and correct submission batch size for your particular project.)

The following describes the steps needed for the matching and registration of records with the EIDR service.

Data Model Alignment Process

The first step in the process is to make sure that any records being registered align with the EIDR data model.

1. Utilizing the defined and required EIDR data fields need to register a record, the EIDR user iteratively reviews their data model to identify and map to EIDR fields.
2. If needed, the EIDR user can request a review of the data model mapping of EIDR, to help validate that the Provider's data model is properly aligned with EIDR.
3. EIDR user supplies sample set of data (for example, 10 complete records) for the fields identified in an Excel template provided by EIDR.

4. EIDR reviews this initial sample to confirm mapping and identify any gaps.
5. If there are missing data fields or other anomalous issues, EIDR user to iterate with EIDR team on how to provide missing fields.

Output: Mapping of EIDR user's data model and practices to EIDR's.

Record Matching Process

The next step in the process is to match the EIDR user's data against the EIDR data.

1. If the user wants to use a third-party matching service:
    a. EIDR will provide a copy of the current registry (in XML or flat file) for external matching.
    b. EIDR user reviews the results to identify records do not exist in the EIDR database.
    c. EIDR user submits these gap records to EIDR for registration.
    d. EIDR will review the provided data to confirm there are no matches and assess the anticipated manual review volumes before registration.
2. If the user wants to use EIDR's fuzzy matching service:
    a. EIDR user submits all of their records to EIDR for preliminary match.
    b. EIDR will review the provided data to confirm the match, no match, and manual review rates before registration.
3. EIDR user then follows the Bulk Registration steps to register records, See steps below.

Output: Gap titles identified and verified for Bulk Registration process.

Bulk Registration Process

After Data Model Alignment and Record Matching have been completed, the registration process can start.

1. EIDR-managed bulk registration:
    a. The EIDR user provides a full set of data fields for the records identified in the output of the Matching Process in a BMR (Bulk Match & Register) template.
    b. EIDR processes them in batches (determined based on the number of anticipated manual reviews), returning the final results once all records have been processed.
2. User-managed bulk registration:
    a. The user works with EIDR Operations to determine a suitable submission schedule for their gap record set (batch size and frequency), based on the number of anticipated manual reviews.
    b. The user processes their records in batches via their own API/SDK integration until all records have been processed.

Output: Gap data added to EIDR database.

NOTE: The EIDR user is then responsible for adding the newly created EIDR content IDs to its data systems.

Bulk Modification Process

1. After Record Matching, if the EIDR user has additional fields such as alternate IDs or more accurate metadata, the EIDR user supplies the EIDR IDs and additional metadata fields.
2. EIDR spot checks metadata for accuracy and alternate IDs for proper granularity.
3. EIDR updates the records.

4 Record Types

EIDR operates four separate ID registries, each with its own record types and data structures:

- Content ID – identifies audiovisual assets of various types.
- Video Services – identifies services that deliver audiovisual content to consumers.
- Party – identifies organizations such as a registrant or producer.
- User – identifies individuals or integrated systems associated with a particular Party.

All EIDR IDs are members of the Digital Object Identifier (DOI) family (ISO 26324). Each EIDR ID begins with a DOI prefix that identifies the registry that issues the ID:

- 10.5240 – Content ID Records: e.g., 10.5240/CA2D-7927-3635-DD8D-7D75-U
- 10.5239 – Video Services: e.g., 10.5237/2135-296E
- 10.5237 – Parties: e.g., 10.5237/2135-296E
- 10.5238 – Users: e.g., 10.5239/FD14-4D40

Video Services, Party, and User registries each contain a single type of record, as indicated by the registry's name.

Party and User IDs are referenced internally by the EIDR access control system, where rights and privileges are assigned to Parties and then inherited by all of the Party's Users. All Registry actions except for Resolution require authentication via User ID and password.

The Content ID registry is more complex than the others and contains a number of different record types, all in service of audiovisual content identification. As noted in ***Introduction to the EIDR Data Model***, the Content ID registry has four fundamental record types:

- Collection – a grouping record such as a Series, Season, or Compilation.
- Abstraction – an abstract work in its most general form, including movies, episodes, and TV specials.
- Edit – creative changes to a work, including both complete versions and clips.
- Manifestation – technical representations and encodings, including language versions ("subs and dubs").

The nature of a Content ID registry record is most strongly determined by the set of relationships it has (e.g., its position in the parent/child record registration tree) and by its referent type (e.g., Series, TV, Web, Movie, etc.).

Content ID Records

The DOI system defines a particular record structure and nomenclature. EIDR extends and adapts this to serve the audiovisual industry. EIDR records are stored in the native EIDR format but can be transposed into the DOI-specific format as needed. See the ***EIDR DOI Mapping Table*** for details of mapping EIDR fields to DOI fields.

This section describes the underlying concepts used to describe content records in the EIDR data model. For more information, refer to ***Introduction to the EIDR Data Model*** and ***Best Practices and Use Cases for Abstraction Records***.

Categorization of Objects

All content objects in EIDR are categorized by their *types* and *relationships*.

TYPE

EIDR has two general kinds of type:

- *Object Type*: Equivalent to the meaning of object *type* in programming languages, which encapsulates the data fields needed for a particular object
- *Record Type*: Describes the fundamental nature of an object, including its level of abstractness

Object Type: This is an extension of the DOI Kernel metadata. When the unqualified word *type* is used in EIDR documents, it refers to these object types.

The content record types are divided into two general classes:

- **Basic Type**: This type covers the minimal possible object. It is sufficient for describing a wide variety of content.
- **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 Compilation, Series, Season, Episode, Edit, Clip, and Manifestation.

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

Structural Type: This is based on a set of four particular structural types provided by DOI. They correspond to increasingly more specific manifestations of a work. Additionally, EIDR has a Record Type that provides a more concise means of referencing an object's fundamental nature.

Structural Type	Use	Record Type
Abstraction	Used for objects having no reality, such as a series collection or the most basic concept of the original work.	Collection: Series, Season, and Compilation records. (Compilations Abstraction, Performance, or Digital, depending on their contents.)
		Abstraction: All abstract works, including both stand-alone (such as Movies) and Episodes.
Performance	Used for items that are particular manifestations or versions of something, such as the Director's Cut of a film or the Welsh-language version of a TV show.	Edit: Edits and Clips (Clips being a special kind of Edit).
Digital	A particular digital manifestation of a work, such as an MPEG-2 encoding of a movie.	Manifestation: Manifestations are further stratified as being Digital or Physical via their Manifestation Class.
Physical	A physical version of an object. EIDR will support this for physical films and tapes in a future release	

Referent Type: In DOI terms, the referent is the item to which the DOI refers and is independent of any particular instantiation. 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.

Referent Type	Use
Series	A collection object for ordered or unordered Seasons and Episodes.
Season	A collection object for ordered or unordered Episodes.
TV	Content that first appeared via broadcast.
Movie	Long-form 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 a work that is 40 minutes or less, such as music videos, theatrical newsreels, or theatrical or DTV cartoon shorts.
Web	Content that first appeared on the Web. This is different from content from elsewhere that has been made available on the Web.
Compilation	A collection of discrete assets such as are found on a home entertainment product, franchise, etc.
Supplemental	This type is for secondary content whose primary purpose is to support, augment, or promote other content. Examples include trailers, outtakes, and promotional documentaries ("making of" pieces.)

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:

Inheritance relationships: The object on which the relationship exists can inherit basic metadata fields from the object to which the relationship refers. Only one inheritance relationship may exist on an object. The Inheritance relationships are *isSeasonOf*, *isEpisodeOf*, *isEditOf*, *isManifestationOf*, and *isClipOf*.

Dependence relationships: The objects to which the relationship refers have a strong bearing on the basic nature of the object on which the relationship exists. This means that the objects referred to in the relationship must be taken into account when checking for duplicates when an object is created or modified. The dependence relationships are *isCompositeOf* and *isCompilationOf.*

Lightweight relationships: There is 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*, *isSupplementTo*, and *isAlternateContentFor*.

There is no structural connection between the Supplemental referent type and the *isSupplementTo* relationship. Although a content record with referent type Supplemental will typically have *isPromotionFor* or *isSupplementTo* to another content record, those relationships can exist on any content record, such as a Clip. Similarly, a Supplemental referent type need not have an *isSupplementTo* relationship to anything.

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 can inherit certain fields from their parent. See the ***Data Fields Reference*** for full descriptions of these fields. Only metadata from Base Object Data can be inherited. Furthermore, an object can only be part of one tree, so it has only a single chain of inheritance. Lightweight and dependence relationships allow records to interact with objects external to their own hierarchy.

This worldview uses standard computer science terminology: ancestors and descendants, root objects and leaf nodes. Items with both ancestors and descendants are called *internal nodes*.

DEPENDENCE

An object may depend on another object in some way by including a reference to it. In such cases 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 Manifestation A refers to Manifestation B by reference, A is dependent on B, and when Composite C includes Clip K, C is dependent on K.

Content Record Creation and Modification

Not all combinations of type, inheritance, and dependence are legal. The validation rules are the normative description of legal and illegal combinations. For more information, see the ***Data Fields Reference***.

In order to reduce complexity, the HTTP API for creating and modifying objects is constrained in several ways, rather than exposing a generic data structure to be filled in. Nonetheless, all legal combinations of type and inheritance can be created and modified using the API, and all legitimate relationships to other objects can be added and removed.

The HTTP `Create()` call and its manifestations in the SDK use `CreationType` as an argument. This table shows the possible uses of `CreationType`.

Creation of Objects		Creation Type
Basic Objects	Referent type can be: TV, Movie, Web, Short, Supplemental	`CreateBasic` or `Basic`
with IsSeasonOf inheritance relationship	Only way to create Season referent type	`CreateSeason` or `Season`
with information for derived type Series	Only way to create Series referent type	`CreateSeries` or `Series`

Creation of Objects		Creation Type
with IsManifestationOf inheritance relationship	Only way to create an object that has IsManifestationOf relationship	`CreateManifestation` or `Manifestation`
with IsEditOf inheritance relationship	Only way to create an object that has IsEditOf relationship	`CreateEdit` or `Edit`
with IsClipOf inheritance relationship	Only way to create an object that has IsClipOf relationship	`CreateClip` or `Clip`
with IsEpisodeOf inheritance relationship	Only way to create an object that has IsEpisodeOf relationship	`CreateEpisode` or `Episode`
with IsCompilationOf dependent relationship	Only way to create an object of Compilation Referent Type	`CreateCompilation` or `Compilation`

A Referent Type of Movie, TV, Short, Web, or Supplemental can be changed to another one of those referent types. Other Referent Types cannot be changed. For example, the Referent Type of a record can change from TV to Movie, or from Web to Supplemental, but Series cannot change to Movie nor can Season change to TV.

Here is the summary of how relationships can interact with each other.

Relationship	Type	Can co-exist with	Can be added after creation	Removable?
IsSeasonOf	Inheritance	Any Lightweight	No	No
IsEpisodeOf	Inheritance	IsCompositeOf and Any Lightweight	No	No
IsEditOf	Inheritance	Any Lightweight	No	No
IsClipOf	Inheritance	Any Lightweight	No	No
IsManifestationOf	Inheritance Dependence	Any Lightweight	No	No
IsCompilationOf	Dependence	Any Lightweight	No	No
IsCompositeOf	Dependence	IsEpisodeOf and Any Lightweight	To Abstractions but not Collections	Yes
IsPromotionOf	Lightweight	Any	Yes	Multiple instances allowed on an object
IsPackagingOf	Lightweight	Any	Yes	Multiple instances allowed on an object
IsAlternateContentFor	Lightweight	Any	Yes	Multiple instances allowed on an object
IsSupplementTo	Lightweight	Any	Yes	Multiple instances allowed on an object

Alternate ID

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, value, and optional relation. For example, an Alternate ID could have a type of ISAN and a value of 0000-0000-61B3-0000-O-0000-0000-2. Proprietary IDs* are supported as well, with an added attribute giving the domain within which the ID is valid. For example, an Alternate ID with the type Proprietary, domain spe.sony.com/MPM, and value E0089786000.

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, such as CITWF, IMDb, or IVA. Studios or other content producers may cross-reference to internal IDs used for other production, distribution, or tracking purposes. EIDR serves as a useful cross-referencing tool for access to a wide variety of external sources of data about each registered asset.

Alternate IDs are also used within the EIDR de-duplication system to help identify similar records that may be described differently thanks to a common Alternate ID.

Aliases and Deletion Model

IDs in DOI registries must be permanent, and the records to which they refer are intended to be permanent. 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, there are three ways of repairing the situation:

- Correct the descriptive metadata associated with the ID record so that it is complete and correct.
- If a duplicate ID already exists, one ID can be aliased to the other. This is used when a duplicate is registered mistakenly, or when a changed understanding of the underlying assets means that they should now be viewed as identical. Both IDs will still exist, but both will resolve to the same underlying EIDR metadata.
- If an ID really must be deleted (because it was a complete error, such as registering test data in the production system) the underlying metadata is removed and the ID is aliased to a *tombstone record*. The ID can still be resolved, which is important if it ever made its way into external systems.

An alias is a simple redirection from one DOI to another. An alias is not intended as a general tool; rather, it should only be used for correcting errors. Records cannot be aliased† if they have any child records or if they are the target of any lightweight relationships.

Because the tombstone object for deleted records is of type `Restricted`, only three of its fields have guaranteed values. Other fields are not used.

Field	Value
ID	10.5240/0000-0000-0000-0000-0000-X

* "Proprietary" is a term of art from the standards world. In this context it means an ID that isn't part of an international standard. It does not imply that the ID is private or confidential.
† Or deleted, since delete is a special case of alias.

Structural Type	Restricted
Resource Name	"EIDR Tombstone Object"

Alias chains: An object may be aliased to an object that is itself an alias. If `followAlias` is `true` when doing a resolution (the normal case) the chain is followed until it ends up at a resolvable record or the chain is more than five levels deep, in which case the registry returns an `aliasContinuation`, which contains the last object reached and the object to which it is aliased. You can continue towards a real object by applying a resolve on the item to which the last object is aliased.

Record resurrection: If an EIDR record is erroneously aliased or deleted, it is possible to reclaim the ID and restore it to active use once again. This is a special administrative procedure. Requests for restoration of an aliased or deleted ID should be carefully considered and only made when absolutely necessary.

Parties

A Party represents an entity such as a Registrant or a producing agent. A User is an individual (or an abstract thing that can be treated as an individual, such as software program performing an automated task).

All Users are associated with a Party.

All Registry requests except Resolve require User authentication for a User, and all requests allow it.

Only Parties have permissions in the system; a User has all the permissions associated with its parent Party.

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.

There is a predefined Party representing EIDR Operations (10.5237/superparty). Only the EIDR operator has the ability to create or modify Parties.

A Party can have one or more of the following roles in its list of AllowedRoles:

- **AssociatedOrg:** These are the organizations involved in the creation of a work: production companies, distributors producing local market edits, post houses encoding or transcoding digital works, etc.
- **Registrant**. The Party that initiated the EIDR record registration, distinct from the Party that created the identified work. The User requesting the creation of a new object must be associated with Party identified as the Registrant of the record.
- **AltIDWriter**. The Party can add or maintain Alternate IDs in any record in the Registry, even if they are not in that record's ACL (Access Control List).
- **MetadataAuthority**: A Party that asserts it has complete and accurate descriptive metadata for the associated asset and agrees to participate in its ongoing maintenance. Need not be the same as the Associated Org(s) or Registrant. This could be the work's original producer, current distributor, or an interested third party such as an archive or film library.

- **EncodingAgent**: A Party that produces encodings/transcodings, generally on behalf of a third party producer or distributor. Only used with Manifestations.
- **ServiceAdmin**: The equivalent of Registrant within the context of the Video Services registry. (All other roles apply to the Content ID registry.)

Additionally, an entity can be registered as:

- **Reader**: If the entity is on an object's ACL (Access Control List), it can read objects and metadata that would otherwise be hidden. This applies only to "in development" objects and provenance data.
- **Writer**: Can read and modify objects, but not create them.

For example, an encoding house that registers new Manifestations is the Registrant for the object. In the strictest sense, they could be the producer as well, but that is not mandated; in this example the rights holder could require that the producer listed for the Manifestation be the same as that for the parent object. The encoding house may also be the EncodingAgent for each item, unless some aspect has been subcontracted to a third party such as a specialist subtitle shop.

Permissions Model

A regular record can be created, modified, aliased, or deleted. An in-development record can also be Promoted. It is also possible to read certain types of administrative information about a record.

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 on the object's ACL in order to be able to create, modify, or read a record's ACL or detailed provenance metadata. (Anyone can view any record or relationship in the Registry along with basic provenance metadata, except for In-Development records, as noted below).

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

- **`Modify`:** Can contain Parties that are of type Registrant or Writer. Required to modify an object.
- **`Delete`**: Can contain Parties that are of type Registrant or Writer. Required to alias or delete an object.
- **`ReadACL`:** Can contain Parties that are of type Registrant, Writer, or Reader. Required to read any ACL.
- **`WriteACL`:** Can contain Parties that are of type Registrant. Required to modify any ACL.
- **`ReadProvenance`:** Can contain Parties that are of type Registrant, Writer, or Reader. Required to read the full provenance metadata.

In-Development records have two more ACLs:

- **`Promote`**: Contains Parties of type Registrant. Required to promote an In Development object to Valid.
- **`View`**: Contains Parties of any type. Required to view In Development objects or have them returned from a query.

Access Control Lists

The following table summarizes possible permissions based on the Role. For a given object, the associated permissions for a particular Party (with a Role) may be more restrictive than what is specified in the table.

Role\Permission	*View*	*Read ACL*	*Read Provenance*	*Modify*	*Delete*	*Write ACL*	*Promote*
Registrant	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Writer	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reader	Yes	Yes	Yes	No	No	No	No
AssociatedOrg	Yes	No	No	No	No	No	No
MetadataAuthority	Yes	No	No	No	No	No	No
EncodingAgent	Yes	No	No	No	No	No	No
AltIDWriter	Yes	No	No	Yes	No	No	No

If a Role is removed from a Party, any ACL that depends on the presence of that role will disallow the action, even if the Party is in the ACL. For example, if the Writer role is removed from a Party, that Party and its associated Users will not be able to modify records for which it was already on the ACLs for Modify, Delete, WriteACL, and Promote, and will not subsequently be able to be added to the ACLs for Modify, Delete, WriteACL, or Promote on any other objects.

In development records, as opposed to v*alid* records, are records that are registered but not officially released. It is not recommended that records be in this state. They have two more ACLs than valid records:

- `Promote`: Entities on this list must be of type Registrant. Entities on this list can promote an In Development object to Valid.
- `View`: Entities on this list can be of any type. Entities on this list can view an in-development object or have them returned from a query.

5 Content Read Operations

There are several ways of getting information about an EIDR ID:

- *Resolution* provides the metadata associated with a particular EIDR ID in various formats, which are:
  - Full
  - Simple
  - SelfDefined
  - DOIKernel
  - Inherited
  - Provenance
- *Traversal* provides information about how the record fits into the object hierarchy, using inheritance, lightweight, and dependence relationships.
- *Queries* return records with metadata matching certain criteria.
- *Matching* returns information regarding potential duplicates for a given candidate record. Matching can be used prior to creating a new record as part of an external de-duplication process to help prevent duplicate registrations from reaching the registry.

Resolution

Resolution provides various views of the metadata associated with a particular ID.

Content 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 referentCreation. Please see *EIDR Data Fields Reference* for full details.

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.

Party and Video Service metadata can be requested as Full or DOIKernel. See the *EIDR DOI Mapping* for details on the latter.

Traversals

Traversals are used for discovering how the content record fits into the graph that represents its relationship to other objects.

Traversals stop at the first object for which the requester does not have read permission. The standard Permission error is used only for permission errors on the ID in the original request.

Traversals always follow aliases.

All traversals return results in depth-first order. With traversals you can perform the following operations:

- Find ancestors
- Get series ancestry
- Get remotest ancestor
- Find descendants
- Get leaf descendants
- Get parent
- Get children
- Get dependent objects
- Get lightweight relationships.

Queries

The query facility takes a set of metadata criteria expressed in XPath format as input and returns `eidr:simpleDataInfoType` for all objects that match those criteria. The simplest query tests a single metadata field. More complex queries can be built up by grouping simple queries together with standard logical expressions.

Queries can be based on the content record's metadata, its modification history (provenance), and its virtual fields. For more information and examples of queries, see Appendix B: Text Processing and Queries.

This query finds all records with "Kung Fu" in the title:

```
(/FullMetadata/BaseObjectData/ResourceName "Kung Fu")
```

This query finds all records that were last modified in 2013:

```
(/ProvenanceMetadata/LastModificationDate = 2013)
```

You can perform a rooted query to get the descendants of a specified object, such as the Episodes for a Series or Season, as shown in the following example using the EIDR command-line tools:

To find all the Episodes registered for a Series with EIDR content ID 10.524/301C-0DFA-B184-5448-BB3E-I as the root object, create the file query-ep.txt containing

```
 (/FullMetadata/ExtraObjectMetadata/EpisodeInfo EXISTS)
```

and run

```
 QueryTool -i query-ep.txt -r 10.5240/301C-0DFA-B184-5448-BB3E-I -t full
```

to see the full metadata for all the episodes of the Series.

Similarly,

```
 QueryTool -i query-ep.txt -r 10.5240/301C-0DFA-B184-5448-BB3E-I -n
```

will tell you how many episodes the Series contains.

This example finds all items that are packaging of a particular record:
`(/FullMetadata/ExtraObjectMetadata/PackagingInfo/ID 10.5240/259A-5359-3425-8B6E-C169-A)`

Other lightweight relationships (such as Promotion) are analogous. You can find anything that is a packaging of something else with
`(/FullMetadata/ExtraObjectMetadata/PackagingInfo EXISTS)`

6 Using Parties

Parties are used as Associated Orgs, Metadata Authorities, and Registrants in Content ID records and as the entities to which Users are attached to give them registry permissions (both registry-wide, such as registration, and record-specific, such as write access to a particular record).

The public Party API allows you to retrieve information about Parties and Users in several ways:

- Search for Parties by name
- Retrieve results from an alphabetized catalog of all the Parties
- Resolve a Party (as EIDR or DOI kernel metadata)
- Resolve a User (as EIDR metadata)

There is a separate administrative API used by the Registry Operator for creating and modifying Parties and Users.

Because there are tens of thousands of production companies, some of which existed to produce a single movie, and because many production companies are related to each other, it is often hard to get reliable information. This in turn can lead to incorrect creation of Party records – for example, "XYZZY Films" may or may not be the same entity as "XYZZY Productions", or they both could be one-time subsidiaries of "XYZZY LLC".

To correct this, EIDR has a process to deprecate duplicate or erroneous Parties. After the request to deprecate a party (containing a preferred party to be used instead) is submitted to the Registry Operator, the deprecated Party is marked Inactive.

In both cases, the names associated with the deprecated Party are added as alternate names to the preferred (correct) Party and uses of the deprecated Party are replaced by the preferred Party. Applications should make sure they do not use (or allow users to use) deprecated Parties when registering or modifying content records.

7 De-Duplication

A good ID is bi-directionally unique; an ID represents a single object, and a single object is represented by only one ID. In order to guarantee universal uniqueness, EIDR content 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.

Overview

The de-duplication module responds to a registry request with one of four outcomes.

- *No Duplicate*: The record submitted is unique.
- *Duplicate:* The record submitted is a duplicate of an existing object in the registry.
- *Potential Duplicate:* It is likely, but not certain, that the record submitted is a duplicate of one or more records in the registry. Synchronous transactions (such as Match), return the match candidate list with confidence scores for the user to review; asynchronous transactions refer the match candidate list to EIDR Operations, so that the user receives a New ID (no duplicate) or Matched ID (duplicate).
- *Rejected:* The submitted record was erroneous or ambiguous and could not be processed further by the registry operator. After correcting the errors or omissions, the record may be re-submitted.

When an attempt is made to register or modify an EIDR content record, the system first decides if the action would result in there being duplicate records in the database. This may occur, for example, if two back-catalogs contain records for the same movie or TV show. Allowing the registration or modification would violate the principal of one record/one ID.

There are sets of rules for determining candidates that match the newly created or modified record, from which a matching system generates scores. Different referent types and relationships have different scoring rules. Scores are computed against two thresholds:

- Low threshold: Anything below this is presumably not a potential duplicate
- High threshold: Anything greater than or equal to this is almost certainly an exact duplicate of the requested registration or modification

The normal operational mode for registration and modification is the asynchronous path. If there is only one candidate and it is above the High Threshold, the system returns the identified duplicate ID in lieu of processing the requested registration or modification. The item returned as the duplicate is probably good enough for the registry to use as the EIDR ID for the requested registration. However, if you are certain that the submitted record is unique, despite its apparent similarity to the identified duplicate, you may provide additional metadata sufficient to disambiguate the record and re-submit the registry request or request a manual review by setting the Operation element's dedupMode flag to "manual".

If there are no candidates above the Low Threshold, then the registration or modification is allowed to complete. Upon successful registration, you receive a new EIDR ID; otherwise, for modification requests, the underlying metadata is modified accordingly.

In all other cases having candidates above the Low Threshold for the asynchronous workflow, the attempted registration or modification is manually reviewed by EIDR to determine if it is an exact duplicate or new item registration.

In the synchronous workflow, which includes basic UI interaction, there are three possible outcomes:

- Success with no matches (new ID issued or existing record modified)
- One high threshold match (existing ID returned, registration or modification not processed)
- One low threshold match or multiple matches of any time (return match list with confidence scores)

When reviewing the results after a failure, the user takes one of the following actions:

- Identifies an exact match and uses that ID. In this case the user has completed the task, though updating the existing record with expanded metadata – including alternate IDs – is recommended practice.
- The metadata must be modified (e.g., fields added) to avoid a match: the user resubmits.
- The metadata is correct: the user submits asynchronously to trigger manual review and returns later for status from the token.
- There is a metadata or validation error: the user fixes the metadata and resubmits.

Matching API

It is possible to obtain de-duplication results without submitting a create or modify request to the registry by using the Match API. This API call is similar to the asynchronous Register API except that no changes are applied to the registry. Instead, the de-duplication results are returned to the user for review. As before, the likely results for each match transaction include No Match (the record should be re-submitted to create a new ID), Match (an existing ID already exists), or Candidate List (possible matches are returned with their confidence scores) to indicate what would happen at registration time.

Tokens and Batches

The EIDR HTTP API has several calls, referred to as *batchable operations*, that can modify the contents of the Registry. These are `Create`, `Modify`, `AddRelationship`, `RemoveRelationship`, `ReplaceRelationship`, `Delete`, `Alias`, and `Promote`.

The EIDR API uses two approaches: single/batch requests and the immediate/async response flag.

- In reality, all EIDR requests are batch requests. What is often called a *single* or *non-batch* request is just a batch containing one request.
- The immediate/async response is a general mechanism, but can only be used for a batch size of one (a single request).

Additionally, the EIDR API uses tokens to track the status of batchable operations.

This section contains a short overview describing how batch/single requests, immediate/async responses, and tokens interact with each other, with examples of the Registry responses for the various combinations.

NOTE: It is important to distinguish the Response returned by the Registry from the value returned by a call through the SDK. In particular, the SDK provides some help with various errors and the registry's occasionally inconsistent error replies, but the objects returned by the SDK have a direct mapping to the Registry Response elements.

Single/Batch

All requests are submitted through the HTTP API as a batch. Batches of one (single requests) are treated somewhat differently from batches with multiple requests.

All the operations in a batch must be the same (for example, all `Create` or all `Modify`). The registry returns an Invalid Request Error for a batch that violates this constraint.

Immediate/Asynchronous

In order to guarantee uniqueness, EIDR sends requests for modifying an object's metadata to the de-duplication system. In most cases this automatically returns a result. If there is ambiguity that cannot be resolved by the software, one of two things will happen:

- If the request is marked as *immediate-response* (synchronous), the registry immediately returns a response to the application, giving details of the potential problems. In some cases, immediate-response requests return more detailed status information than asynchronous requests.
- If the request is not marked as immediate-response, it is sent for manual de-duplication. Registry operators make a decision, which is returned to the application. This process is not real-time, and these requests are usually referred to as asynchronous.

Immediate response applies ***only*** to single requests, and all multiple-request batches are non-immediate (asynchronous). If an application requests immediate response for a batch of more than one item, the registry returns an Invalid Request error. For example:

```
<Response xmlns="https://www.eidr.org/schema" version="2.6.0">
 <Status>
  <Code>3</Code>
  <Type>invalid request</Type>
 </Status>
</Response>
```

Tokens

Every batchable request generates a token for the request; a multi-item request additionally generates a token for each operation in the batch. This is done with two kinds of tokens

- *Operation tokens*, which refer to individual Create, Modify, etc. requests and are returned in the `/Response/RequestStatusResults/OperationStatus/Token` XML element.
- *Batch tokens*, which refer to the status of a batch request. These are returned in the `/Response/RequestStatus/Token` element.

In addition, the user can assign a unique User Token to any batchable operation, generally the user's internal ID for the associated transaction. This may simplify certain integrated system workflows, since the user's system will not have to store the EIDR token.

Information is extracted from tokens with the `StatusLookup` request. Operation Tokens have detailed information about the status of an individual request (for example, a single Create or Modify). Batch tokens have information about the status of the batch and any available information about the individual items within the batch. This information includes the Operation Token and current state for each item in the batch.

Batches with a single item generate only a single token. This is treated as an Operation Token whenever information relating to it is returned from the Registry (for example, when it is initially generated, and when it is requested via StatusLookup).

Operation Tokens

The `/RequestStatusResults/OperationStatus/Status` element will not change once it has reached a terminal state. Anything other than Pending is a terminal state. The `/RequestStatusResults/OperationStatus/Status/Code` is a numeric value from 0-5 with corresponding `OperationStatus/Status/Type` strings. These elements, as well as the codes and types for other fields, are defined in api-common.xsd.

OperationStatus Code	OperationStatus Type
0	success
1	duplicate
2	pending
3	authorization error
4	validation error
5	other error
6	rejected

Records can be re-submitted to obtain a different result after making suitable changes:

- Duplicate Error (change the metadata in the request, the metadata of the object(s) that were found as duplicates, or the de-dupe mode).
- Authorization Error (change the credentials in the request, ACL of any objects involved, or the roles allowed to the requester).
- Validation Error (change the metadata in the request or the metadata on related object(s) that caused the problem).
- Other Error is returned for various transient problems (such as bad communication with the de-duplication system) and can be retried without necessarily changing the submitted data. Since it may reflect some other error, and *transient* does not necessarily mean short-lived, some caution should be used – if this error is returned a second time, it may not be productive to try it a third time without first contacting EIDR Operations.

Pending us the only non-terminal state for an operation token.

Batch Tokens

For batches containing more than one item, once a batch has passed top-level authentication, syntax checking, etc., there are three possible states:

- 1 (batch received) means that the batch has passed the preliminary validation and is being turned into individual requests. No further information is available at this point.
- 2 (batch queued) means that the individual requests have all been submitted. In this state the individual tokens and the current state for each are returned when you call `StatusLookup` with the batch token.
- 3 (invalid batch) can result from a bad user token. This can also result from abnormal operation of the Registry, which should be reported to EIDR support.

Batch queued and *invalid batch* are the terminal states for a batch token.

Scores

Scores can be returned in the response to immediate-mode requests to indicate how close any duplicate items are to the requested registration. Scores are valid ***only*** for immediate-mode (synchronous) requests. If they are present in the response to an async request, they should be ignored.

Polling

You must periodically poll that status of a Token using StatusLookup until it reaches one of the possible end states.

For a Batch Token you can extract all the Operation Tokens once the batch has reached the "batch queued" state and manage them all individually, or you can poll on the Batch Token, dealing with each Operation Token as it reaches an end state or after all of them have reached an end state. The former is usually preferable, since you do not have to continually poll on the Batch Token; the latter is less efficient but may be preferable when you do not want the complexity of managing multiple tokens. Using the Web UI, you can only submit one record at a time but can poll any token. If you search for a batch token you will see the operation's token results, and you can look up each operation token separately.

8 Content Create and Modify Operations

This section provides information on registration workflows, modifying records, alias operations, and delete operations.

Registration Workflows

This section contains a description of synchronous and asynchronous registration workflows, including the usage of the user interface and automated processes.

This is how people are encouraged to use the registry for new Collection and Abstraction registrations (e.g., episodes in a season, movies, one-time-only television, new series). For Edits and Manifestations, any difference in metadata is probably distinguishing and does not normally require manual review (for example, a special version made for international release as opposed to domestic, MPEG4 instead of MPEG2).

In the case of someone performing a manual registration with the user interface, the process is somewhat transparent. If an asynchronous workflow requiring manual review returns a list of potential candidates, someone using the Web UI can review the list of candidates and choose one that matches, make their metadata distinct and resubmit, or request manual review by EIDR staff.

Synchronous Workflow

The synchronous workflow results in an immediate response, which could include a candidate list requiring manual review. If there are no candidates (nothing above Low Threshold), the result is immediate success with a new registration. If there is a system error, validation error, or record rejection, the user must fix the submission and resubmit. Otherwise, the system returns either a high single match or a list of candidates for consideration.

When reviewing de-duplication results, the user takes one of the following actions:

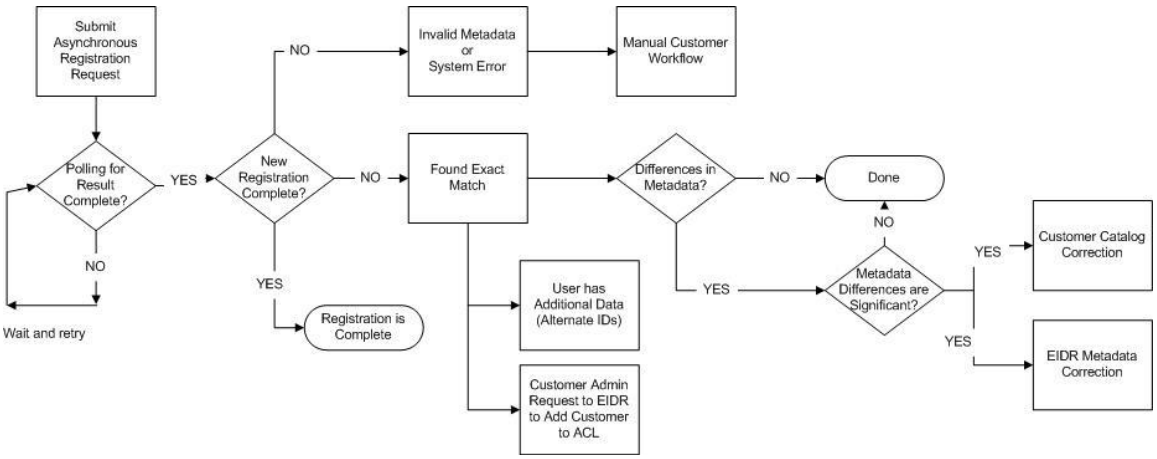
- Identifies an exact match. In this case the user has completed the task, though updating the existing record with expanded metadata – including alternate IDs – is recommended practice. If the user does not concur with the exact match identified, the record can be submitted for manual review by setting the dedupMode flag on the Operation element and re-submitting asynchronously.
- The metadata must be modified: the user resubmits synchronously.
- The metadata are correct: the user submits asynchronously to trigger manual review and returns later for status from the token.
- There is a metadata or validation error: the user fixes the metadata and resubmits. There is always the chance of a system error as well, though unlikely.

Asynchronous Workflow Tools and API

API 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.

API-based Asynchronous Workflow

The following diagram illustrates the asynchronous registration process using an automated API-based workflow:



An API-based application submits asynchronously and polls the result for a status of complete or incomplete. If the status is incomplete, the application waits and polls again. If the request is resolved automatically it will likely be returned within 10 seconds. It is advised to program a "back off" for progressively slower polling leading up to a one business day delay for manual review results.

If there is invalid metadata or a system error, a manual workflow may be used, registering in the UI may be attempted, or code logic errors may be identified.

There are three possible outcomes:

- An exact match is found. Compare the metadata for differences. If there is no difference, the process is complete (done). Otherwise, determine if the differences are significant. If they are not, the process is complete (done). If EIDR has incorrect or incomplete information, then the EIDR registry must be updated via a modification workflow.
- If the user has additional data such as an internal studio identifier (Alternate IDs) to be placed on the EIDR record, the user should initiate a modification workflow.
- Process modifications are needed. The EIDR user makes an administrative request to EIDR to add the User to the ACL. After the User is added to the ACL, the UI, API, or an Admin request to EIDR may be used to modify records.

In the case of new registration, there should be no corrections to the metadata and alternate IDs are part of the new registration request. The process is complete (done).

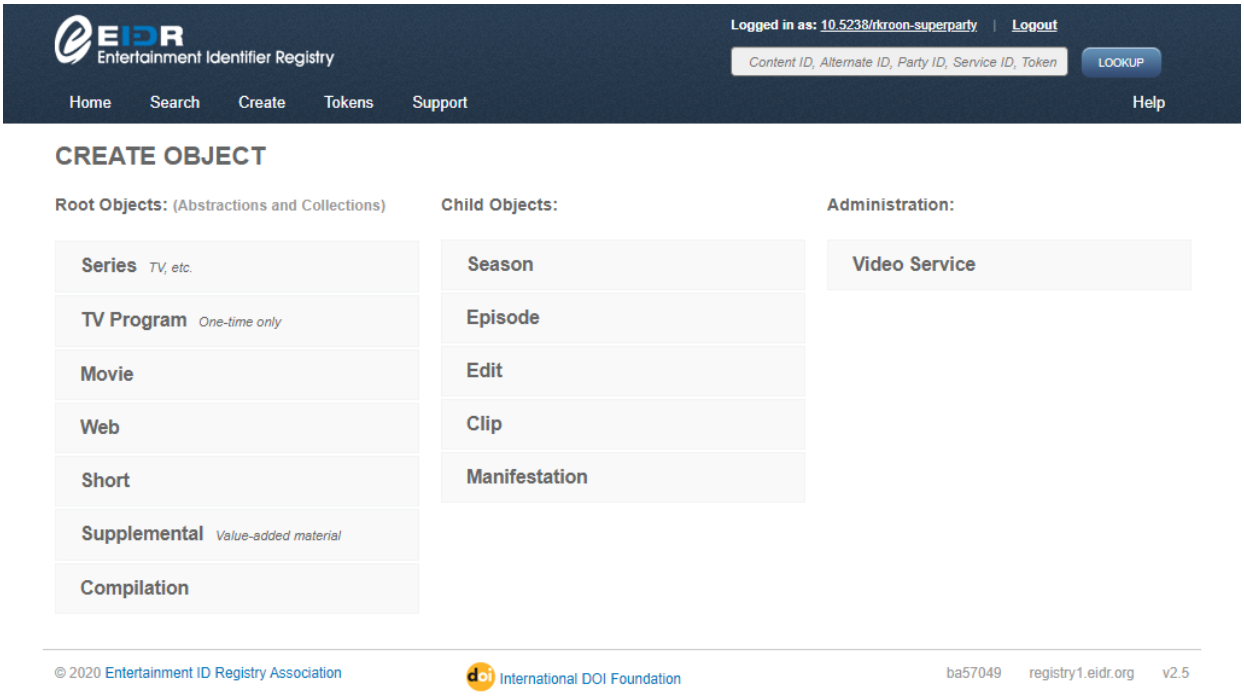
User Interface Handling

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

- Resolve an ID: View its metadata and relationships
- Search for records based on one or more know object attributes
- Create objects, with special screens for common cases (such as OTO)
- Modify objects
- Add or remove relationships for a selected object

- Create objects similar to or related to an existing object in the registry
- Check status of submitted registrations.

This example shows the Create screen of the Web UI with the list of Root and Child Objects that one can register in the EIDR system:



Modifying Records

Modifying objects is done by retrieving current information on the object, modifying it, and re-submitting it, which must pass validation and de-duplication. There are APIs for retrieving object metadata and modifying an object.

If a data element is changed in a parent object, then that same element is changed in any child objects that inherited their values from the parent.

Objects having a status of “in development” are not checked by the de-duplication service. They are sent for de-duplication when they are promoted to a status of ”valid”.

Here are the permitted ways to add relationships after object creation:

The dependent relationship IsCompositeOf and the lightweight relationships (IsPromotionOf, IsPackagingOf, IsAlternateContentFor, and IsSupplementTo) can be added to any valid object at any time after original creation.

Any relationship that can be added after creation can be removed.

Modifying objects has two components:

- Changing Base Object Data only (Basic creation type) while preserving all Extra Object Metadata.

- Changing metadata for the creation type with which the object was originally created (Base Object Data plus any applicable Extra Object Metadata).
- Changing metadata for relationships which have been added to the object.

This table gives some examples of legal modification bases for records as they are created and modified. This table is informative, and is not exhaustive.

9 Registry Operation Status Codes

The following Status Codes apply to all operations:

Status Code	Status Type	Note
0	success	Indicates that the API request succeeded.
1	system error	Should be reported to EIDR support
2	registry in read-only error	Should be reported to EIDR support unless this Registry is a mirror or is scheduled to be read-only.
3	invalid request	An API (URI) that does not exist including missing a required parameter. Remember that parameters (such as type=Simple) are case sensitive. Could also be POST data that is syntactically invalid such as missing required headers or if the end-of-line characters are not CR-LF.
4	authentication error	Invalid credentials including an Inactive account.
5	authorization error	The operation requires credentials. Or the credentials provided are not authorized to perform this operation. Check with EIDR support about this operation.
6	bad token error	There is a problem with the token ID such as one that is not syntactically valid or does not exist.
7	bad query error	There is a problem with a content record query. This could include a typographical error in an EIDR field name.
8	bad id error	There is a problem with the content ID such as one that is not syntactically valid or does not exist.
9	syntax error	Invalid XML in a query or write operation. Examples: an incorrect namespace declaration; an element not closed; or incorrect case for an enumerated value.
10	result too long	A result was too large to fit in a HTTP response. This can be caused by requesting too large a page size in queries.
11	duplicate party	An Administration API error
12	duplicate user	An Administration API error
13	bad party	There is a problem with the Party ID such as it does not exist

Status Code	Status Type	Note
14	bad user	There is a problem with the User ID such as it is syntactically invalid or does not exist.
15	all valid	An Administration API error
16	wrong group	An Administration API error
17	Invalid	An Administration API error
18	no parent	The object of a GetParent request is itself the root of a content record tree.
19	no children	The object of a GetChildren request is itself a leaf of a content record tree.
20	has dependents	The content record cannot be deleted because it has dependent relationships.
21	duplicate service	The same Video Service already exists. (The Video Service registry does not have a full de-duplication system, so the registry reports any duplicates it finds directly.)
22	bad service	Invalid Video Service ID.
23	compatibility error	The operation cannot be supported with the requested value in the EIDR-Version header.

When the Request is a content Operation, the Response also includes an Operation Status code as described in the Operation Tokens section. The "validation error" usually includes a Details field. Examples of Details text include:

Details Text	Note
ID must be an identifier of a valid record	An operation was attempted on an invalid ID. For example to Modify an aliased record or deleting a record that is already deleted.
Must be a <Type> object	Attempt to modify an object with an incompatible data type. See the Modifying Records section.

Appendix A: Text Processing and Queries

Field Rules

There are three kinds of matches:

- Token Match – the check is made for the existence of the tokens in any order in the candidate string. This uses the `<field> <string> <string>` syntax.
- Exact Match – the check is done for the existence of an ordered sequence of tokens in the candidate string. This uses the `<field> "<string>"` syntax.
- Complete Match – checks for the exact existence of the exact query string or not. This is generally used for fields containing controlled vocabulary or IDs, but can also be used on some text fields. For a non-tokenized field, the query string and the field must be identical. A tokenized field must be identical to the tokenized query string. This uses the `IS` and `ISNOT` operators.

Controlled vocabulary fields have punctuation replaced by a space, and are then tokenized.

Field	Match type	Normalize
ResourceName	Token	Yes
	Exact	Yes
AlternateResourceName	Token	Yes
	Exact	Yes
Any field called DisplayName	Token	Yes
	Exact	Yes
VirtualField (Full and self-defined) See description of Virtual Fields.	Token	Yes
	Exact	Yes
Description	Token	Yes
	Exact	Yes
HouseSequence	Complete	No
RegistrantExtra	Complete	No
RegistrantPrivate	Complete	No
AlternateID	Complete	No
FindPartiesByName	See function definition	Yes
FindPartiesFromCatalog	See function definition	Yes

Simple Queries

The metadata fields to be tested are represented with a subset of XPath notation that supports only complete paths to elements and attributes. (For more information on XPath, see https://www.w3.org/TR/xpath20/.) The XPath used in query requests can be based on:

- `/FullMetadata`, which is of type `fullObjectInfoType`: This queries across the objects' inherited metadata.
- `/ProvenanceMetadata`, which is of type `provenanceInfoType`: This queries across the objects' provenance metadata.

In these examples the parentheses are not strictly necessary, but improve legibility. Note that XML, at the protocol level, requires escaping of special characters (<, >, etc.), although procedural implementations of the API may hide that from the application.

This query finds all objects longer than 20 minutes and shorter than 40 minutes:

```
(/FullMetadata/BaseObjectData/ApproximateLength > PT20M00S) AND
(/FullMetadata/BaseObjectData/ApproximateLength < PT40M00S)
```

This finds all records modified in 2010 and not modified since. (See below for the precision of date comparisons.)

```
(/ProvenanceMetadata/LastModificationDate = 2010)
```

There are several kinds of simple queries, not all of which are applicable to all fields.

Exact Value: These queries use IS and ISNOT. They are applicable to

- Fields containing DOIs
- Fields containing controlled vocabulary
- Certain text fields.

NOTE: For a non-existent field, ISNOT returns TRUE.

Exact Value-language: This special case of Exact Value fields for language fields uses IS and ISNOT and behaves as follows:

- If only a pre-dash component is supplied in the query, it matches anything with that prefix.
- If the language code in the query has a – in it, it only matches another field that is exactly the same.
- Examples:
  - es matches objects that have es, es-ES, and es-419
  - de-CH matches de-CH, but not de or de-DE.

Order: Queries can be done using comparisons (<, <=, >=, >) as well as equality and inequality (=, <>, !=) for fields that contain:

- Integers
- Dates
- Durations.

Existence: The existence of a field can be queried. This is useful for optional elements that represent large optional sub-blocks (e.g. the subtitle tracks of a Manifestation).

For example, this finds all objects that have information about separately encoded subtitles:

(/FullMetadata/ExtraObjectMetadata/ManifestationInfo/Digital/Track/ Subtitle EXISTS)

Text Matching:

- Text queries are case-insensitive.
- Both the text in the query string and the text stored in the registry are generally processed into tokens before matching. Tokenization consists of one or more of the following steps.
  - Normalization: Sequences of whitespace are collapsed into a single space; some punctuation is converted to spaces; and some punctuation is removed (causing concatenation of the string before it with the string after it). This gives a series of tokens.
  - For Description fields only, two filters can be applied to the tokens that result from normalization:
    - Stop words (small, common words, such as "the" or "in" in English or "la" and "en" in Spanish) are filtered.
    - Words are stemmed; stemming removes plurals, turns inflected words into the appropriate root, and so on.
- Strings represented using the Latin alphabet can be searched with or without diacritic significance. ASCII-based searches ignore diacritic marks ("ü" is equivalent "u"), while non-ASCII searches treat characters with different diacritics as distinct.

Search Expressions

There are two kinds of text queries:

- The form **<field> <string1>…<stringN>** is true for any field that has one or more of the strings. It is equivalent to **<field> <string1> OR <field> <string2> OR ... OR <field> <stringN>**
- The form **<field> "<string>"** is true for any field that has exactly **<string>** in it. **<string>** is tokenized before the comparison. Stated another way, the token sequence generated by **<string>** must appear exactly in **<field>**. The tokenization rules applied to applied to **<string>** are those applied to **<field>**.

The grammar for query expressions is:

```
<expression> ::= <term>
               | <expression> OR <term>
               | <expression> AND <term>
               | NOT <term>
               | ASCII (<expression>)

<term>       ::= <field> EXISTS
               | <field> <string> <string>*
               | <field> "<string>"
               | <field> IS "<string>"
               | <field> ISNOT "<string>"
               | <field> <logop> <value>
               | ( <expression> )
```

NOTE: * is the equivalent of EBNF {} and "<term> || NOT <term>" could be EBNF "[NOT]<term>"

```
<field>      ::= legal xpath attribute
  | legal xpath element

<value>      ::= number | date | time | duration
<logop>      ::= = | <> | < | <= | > | >= | !=
```

ASCII Searches

Using the ASCII operator in a query string changes the way Latin alphabet-based text strings are compared so that characters with and without diacritic marks are evaluated identically by mapping them all to their ASCII equivalents. The mapping is based on Unicode NFKD decomposition plus the Latin supplement (Latin-ASCII.xml) from the Unicode Common Locale Data Repository. When searching in this mode, "ü" is equivalent "u" and "ł" is equivalent to "l". This means that in most cases, ASCII versions of Latin content titles no longer need to be created manually.

To search in this mode include an ASCII modifier to one or more query Expression clauses:

```
ASCII((/FullMetadata/BaseObjectData/Credits/Actor/DisplayName Martín) OR
(/FullMetadata/BaseObjectData/Credits/Actor/DisplayName Jose))
```

This would find actors named Martin or José.

NOTE: ASCII searches do not automatically account for locale-dependent forms, such the "ü" in German which may be represented as "ue" in English or Latin transliterations (Romanization) of non-Latin scripts such as Cyrillic, Chinese, or Arabic, which must still be produced manually.

Notes and Examples

- The types on each side of a `<logop>` must be compatible.
- Wildcards are not currently supported; normalization and stemming cover the problems for which wildcards are generally used.
- Comparison operations for dates and times truncate to the lowest precision in the expression.
- Although ranges are not directly supported, they can be implemented using two simple queries combined by AND.
- Fields that contain controlled vocabulary are tokenized, with punctuation characters removed.
- The empty string matches nothing, rather than everything.
- For non-existent fields, all ISNOT comparisons evaluate as True. For example, if there is no CountryOfOrigin field, **`(/FullMetadata/BaseObjectData/CountryOfOrigin ISNOT fr)`** is True.
- `IS` and `ISNOT` apply to Value, Value-language, and Text fields.
    - For Value fields, they are useful for testing for controlled vocabulary words, equality of DOIs, and equality of non-tokenized fields such as HouseSequence, AlternateID, and the various private data fields.
    - For Value-language, they are used as described above.

    - For text fields, the field and the string have the same tokenization rules applied (see Appendix B: Text Processing and Queries for individual fields).

- Comparisons are done to the precision of the least precise argument. For example, a date field containing 2010 is >=, <=, and = to 2010-10-10. Using >=2012-01-01 would return the records in 2012 and later.
- Some applications may want to do queries across only metadata on the object itself, as opposed to the full metadata. This can be useful for applications whose main purpose is dealing with the metadata, rather than dealing with the objects defined by the metadata. This can be done by doing a regular query, calling `Resolve()` to return only self-defined metadata, and then examining those results.
- As an example, imagine a Registry that has objects with the following titles in the ResourceName fields:
    - Batman: The Dark Knight
    - Knight of Dark Stories
    - Dancing In The Dark
    - Darkness At Noon
    - Darkness Waits
    - The Ghost and The Darkness
    - Shanghai Knights
    - Shanghai Noon
    - Sinbad: The Battle of the Dark Knights
    - First Knight

Querying on **`/FullMetadata/BaseObjectData/ResourceName`** (abbreviated **`field`** in the table) gives these results:

Expression	Results	Notes
`field Dark`	Batman: The Dark Knight Knight of Dark Stories Dancing In The Dark Sinbad: The Battle of the Dark Knights	Anything with “Dark”.
`field "Dark Knight"`	Batman: The Dark Knight	Anything with exactly the sequence “Dark Knight”. Sinbad: The Battle of the Dark Knights is not included because titles are not stemmed.
`field Dark Knight`	Batman: The Dark Knight Knight of Dark Stories Dancing In The Dark Sinbad: The Battle of the Dark Knights	Any title that has “Dark” or “Knight”.

Expression	Results	Notes
	First Knight	
`field Knights`	Shanghai Knights Sinbad: The Battle of the Dark Knights	Any title with "Knights".
`(field Dark) AND (field Knight)`	Batman: The Dark Knight Knight of Dark Stories	Any title with both "Dark" and "Knight", in any order and any position.
`(field Dark) AND NOT (field The)`	Knight of Dark Stories	Sinbad: The Battle of the Dark Knights is not included because comparison is case-insensitive.

NOTE: The `ISNOT`, `NOT` and `<>` operators can be inefficient when applied globally.

Example Queries

Finding Types of Objects	
Find all Series, 1 Also works with "Season"	(/FullMetadata/BaseObjectData/ReferentType Series)
Find all Series, 2 Also works with SeasonInfo, EpisodeInfo, ClipInfo, CompilationInfo, CompositeInfo, ManifestationInfo, PackagingInfo, PromotionInfo, AlternateContentInfo, SupplementalContentInfo	(/FullMetadata/BaseObjectData/SeriesInfo EXISTS)
Find all records	(/FullMetadata EXISTS)
Find all root objects. This is done by checking for the absence of any relationship that requires a Parent.	(NOT ((/FullMetadata/ExtraObjectMetadata/SeasonInfo EXISTS) OR (/FullMetadata/ExtraObjectMetadata/EpisodeInfo EXISTS) OR (/FullMetadata/ExtraObjectMetadata/EditInfo EXISTS) OR (/FullMetadata/ExtraObjectMetadata/ClipInfo EXISTS) OR (/FullMetadata/ExtraObjectMetadata/ManifestationInfo EXISTS)))

Registrant and AssociatedOrg	
Find all "in development" records for a Registrant.	(/FullMetadata/BaseObjectData/Administrators/Registrant 10.52337/ABCD-EF01) AND (/FullMetadata/BaseObjectData/Status Dev)

Registrant and AssociatedOrg	
Find all valid records with a particular AssociatedOrg.	(/FullMetadata/BaseObjectData/AssociatedOrg@organizationID 10.52337/ABCD-EF01) AND (/FullMetadata/BaseObjectData/Status valid)
Find all valid records with one or the other of two AssociatedOrg IDs. Generalization is left to the reader.	((/FullMetadata/BaseObjectData/AssociatedOrg@organizationID 10.52337/ABCD-EF01) OR (FullMetadata/BaseObjectData/AssociatedOrg@organizationID 10.52337/2345-6789)) AND (/FullMetadata/BaseObjectData/Status Valid)
Find things registered by the EIDR Operations.	(/ProvenanceMetadata/Administrators/Registrant IS 10.5237/superparty)
Find things not registered by the EIDR Operations.	(/ProvenanceMetadata/Administrators/Registrant ISNOT 10.5237/superparty)

Looking for Possible Data Quality Problems	
All Items with English title and non-English primary language. The records may be correct (*The Hangover* was released as *Very Bad Trip* in France) but is quite often not, so it is worth investigating, especially for bulk registration.	Method 1: (/FullMetadata/BaseObjectData/ResourceName@lang en) AND (/FullMetadata/BaseObjectData/OriginalLanguage ISNOT en) Method 2: (/FullMetadata/BaseObjectData/ResourceName@lang en) AND NOT (/FullMetadata/BaseObjectData/OriginalLanguage en)
Find everything from before 1936 that is not a Movie or a Series.	(/FullMetadata/BaseObjectData/ReleaseDate <= 1936) AND (/FullMetadata/BaseObjectData/ReferentType ISNOT Movie) AND (/FullMetadata/BaseObjectData/ReferentType ISNOT Series)
Bad Season End date These can creep in if an export program uses a silly default when there is no date in the database. Change SeasonInfo to SeriesInfo	(/FullMetadata/ExtraObjectMetadata/SeasonInfo/EndDate > YEAR) *Where YEAR is some year in the far future.

Looking for Possible Data Quality Problems	
for bad Series end dates. You can also generate queries like this for checking consistency for series, using either tools and scripts or the SDK. -- Do a Full resolution -- Extract the end date -- Construct the query, using the series as the root of the query	
Find things with EIDR Operations as AssociatedOrg.	(/FullMetadata/BaseObjectData/AssociatedOrg@organizationID 10.52337/Superparty)
Statistics (The –n flag in QueryTool is useful here)	
Find all records modified since 31 December 2010. Use this to do incremental backups, setting the date to be the day before you started the last one (to avoid race conditions and time zone issues).	(/ProvenanceMetadata/LastModificationDate >= 2010-12-31)
Find any record that was submitted in February, 2013.	(/ProvenanceMetadata/CreationDate >= 2013-02-01) AND (/ProvenanceMetadata/CreationDate < 2013-03-01)
Find all records registered by Registrant 10.5237/ABCD-EF01 in August, 2012.	(/ProvenanceMetadata/Administrators/Registrant IS 10.5237/ABCD-EF01) AND ((/ProvenanceMetadata/CreationDate >= 2012-08-01) AND (/ProvenanceMetadata/CreationDate < 2012-09-01))

Language-specific Filtering

NOTE: Language-specific filtering applies only to registry queries operating on the Base Object Data Description field. It does not apply to queries on other data elements and does not apply to de-duplication.

There are language-specific lists for English, French, Spanish, Italian, and German for:

- Punctuation that turns into spaces
- Punctuation that collapses two words together
- Stop words that get filtered out

Language-specific rules are given in the table below. If a field has a language attribute, then the language-specific rules are applied; otherwise, English rules are applied. The query string is processed based on the language field in the queried field; if there is no language attribute, English rules are applied.

NOTE: The dash in language fields (e.g. de-CH) is not removed.

Language	Spacing Punctuation	Collapsing Punctuation	Stop Words
English	. , ; : ^ & ! + - = ( ) [ ] { } < > ~ # $ / * @ € £ ? " (double quote) - (hyphen)	' (single quote) ’ (apostrophe)	a, the, this, that, these, some, is, are, and, or, but, so, as, at, by, of, on, for, in, into, to, with, I, you, he, she, it, we, they, them, its, theirs
French	. , ; : ^ & ! + - = ( ) [ ] { } < > ~ # $ / * @ € £ ? " (double quote) - (hyphen) « »	' (single quote) ’ (apostrophe)	un, une, le, la, les, l, ce, ces, c, de, du, des, d, est, sont, a, ont, ne, pas, n, et, ou, mais, que, qui, qu, à, aux, sur, dans, en, par, avec, y, il, elle, ils, elles, lui, leurs, son, sa, ses, leur
Italian	. , ; : ^ & ! + - = ( ) [ ] { } < > ~ # $ / * @ € £ ? " (double quote) - (hyphen) « »	' (single quote) ’ (apostrophe)	ad, al, allo, ai, agli, all, agl, alla, alle, con, col, coi, da, dal, dallo, dai, dagli, dall, dagl, dalla, dalle, di, del, dello, dei, degli, dell, degl, della, delle, in, nel, nello, nei, negli, nell, negl, nella, nelle, su, sul, sullo, sui, sugli, sull, sugl, sulla, sulle, per, tra, contro, lui, lei, noi, loro, suo, sua, suoi, sue, lo, la, li, le, gli, ne, il, un, uno, una, ma, ed, se, perché, anche, come, dov, dove, che, chi, cui, non, più, quale, quanto, quanti, quanta, quante, quello, quelli, quella, quelle, questo, questi, questa, queste, si, a, c, e, i, l, o, sono, è
Spanish	. , ; : ^ & ! + - = ( ) [ ] { } < > ~ # $ / * @ € £ ? " (double quote) - (hyphen) « » ¿ ¡	' (single quote) ’ (apostrophe, but it is not used in Spanish)	un, unos, una, unas, el, los, la, las, este, esta, esto, estos, estas, ese, esa, eso, esos, esas, es, son, está, están, hay, y, o, pero, de, en, para, como, con, por, sobre, el, ella, ellos, ellas, se, su, sus, suyo, suya, suyos, suyas
German	. , ; : ^ & ! + - = ( ) [ ] { } <	' (single quote)	ein, einer, eine, eines, einem,

Language	Spacing Punctuation	Collapsing Punctuation	Stop Words
	> ~ # $ / * @ € £ ? " (double quote) - (hyphen) „ “ ” “ « »	’ (apostrophe) [**NOTE:** This means that an apostrophe at the end of a word is dropped, and one in the middle of a word collapses the two parts together.]	einen, der, die, das, den ist, sein und, oder durch, als, von, mit, für, am, in, aus er, sie, es, sie ihn ihm, ihr, ihnen sein, siene, ihre

Glossary

Abstraction - This represents the underlying intellectual property of an audio-visual work or the concept or idea of the work. An abstraction does not represent any particular performance and may have several child objects that are different Performances or Manifestations. For a Movie this would be the root-level record. For episodic television, this applies to the Episodes. This does not apply to the following Referent Types: Series, Season, Compilation, Clip, Edit, or Manifestation.

A Structural Type inherited from the DOI standard that is found in Base Object Data. Applies to all root records and Episodes, regardless of their Referent Type. The EIDR Structural Type is equivalent to the DOI creationStructuralType type.

ACL (Access Control List) – The list of Parties who have modify rights on a particular registry record. The Superparty is implicitly on all ACLs. The record's original Registrant is given write access when the record is created. Other Parties may be granted write access at a later date. Parties with the AltIDWriter role can modify the Alternate ID information on any record, even if they're not on the record's ACL.

alias – Deprecating a duplicate EIDR ID so that it resolves to a different EIDR ID. When duplicate records are found, one is aliased to the other. The aliased ID is still valid for all external uses (the ID is permanent). Within the registry, it may only be resolved (in which case, it returns the survivor's ID and metadata if follow alias is set). Its metadata may not be modified and it may not be given new child records. Those functions are limited to the surviving record.

asset – The identified object or target of an EIDR ID.

Associated Organization – One or more entities that have a role in the creation of an asset. The role can be producer, distributor, broadcaster, encoder, editor, or other. This does not imply any ownership or rights to the asset or record.

asynchronous mode – If records cannot be processed within a defined timeout interval, then the registry returns a transaction ID (Token) instead. Processing continues and the user may obtain the transaction's state via the Token.

automatic match; auto match – A duplicate match identified by automated processing without manual review.

Base Object – Describes a content record that is created solely with base object data. Root instances of Movie, TV, Web, Short, and Supplemental Referent Types can be defined solely with base object data.

base object data – A set of fields that are common to all Content ID records. Some base object data fields can be inherited by child records.

batch mode – Submitting multiple, similar transactions as part of a single operation. Each individual item in the batch is processed separately.

candidate pool – A subset of the entire registry containing the records that will each be compared to the submitted record and scored according to the defined rule set.

child – A record that has a parent object in the registry's single inheritance tree. Child objects may inherit certain metadata from the parent.

Clip – A data type for a single, contiguous fragment of an asset. A sub-type of the Edit Record Type.

Collection – A Record Type that identifies records that contain other records: e.g., Series are collections of Seasons and Episodes; Seasons are collections of Episodes; and Compilations are collections of Abstractions, Edits, or Manifestations.

Compilation – The data type and Referent Type for assets that are composed of discrete multiple assets such as are found on a home entertainment product. Part of the Collection Record Type.

Composite – A single continuous item composed of a sequence of other items in whole or in part. Composite data can be added to any record with an Abstraction Record Type.

Content ID – The EIDR ID for an audiovisual asset record, identified by the DOI prefix 10.5240.

Content Record – A record in the EIDR system that represents an audiovisual asset. This includes movies and television programs.

de-duplication – The process of ensuring that an EIDR record identifies a unique asset. All changes to the registry (create and modify) go through automated de-duplication. If the automated system cannot make a final determination, then the transaction goes through manual de-duplication. (For synchronous transactions, manual de-duplication is performed by the user; for asynchronous transactions, manual de-duplication is performed by EIDR Operations.)

De-Duplication Service – An internal service within an EIDR System that helps prevent the registration of duplicate records. De-Duplication is automatic where possible.

delete – Once issued, EIDR IDs are never deleted. Generally, an ID is unwanted because it was a duplicate, and is aliased to a survivor. In rare cases, the ID should never have been issued and cannot be associated with any valid record. In lieu of deletion, these IDs are aliased to the tombstone record for that registry.

dependence relationship – A non-inheritance dependency between two records: Compilation (a Collection record that identifies a bundle of Abstraction, Edit, or Manifestation records); Composite (an Abstraction record that identifies some number of source works that provided content found in the Composite work); or Manifestation (when a Track includes an External Track Reference).

Derived Types – Objects of derived type include extra object metadata (in addition to base object data). The derived object types are: Series, Season, Episode, Clip, Compilation, Edit, and Manifestation.

Digital – This Structural Type applies to Manifestations.

DOI – The Digital Object Identifier standard. This identifier is the building block of the DOI® System. EIDR is based on DOI.

DOI Kernel – A resolution type within the EIDR Registry. This format provides a subset of EIDR fields in DOI format.

DOI Name – A unique identifier within the DOI System. This ID consists of a prefix (namespace) and a suffix, separated by a forward slash (/). All EIDR IDs are DOI Names.

DOI prefix – A unique identifier within the DOI System that identifies a particular namespace. The current EIDR namespaces have DOI prefixes of 10.5237 (Parties), 10.5238 (Users),10.5239 (Video Services), and 10.5240 (content records). In a DOI ID's canonical form, the DOI prefix is separated from the suffix by a slash ("/").

DOI Proxy – A service operated by the DOI Foundation for DOI resolution, including EIDR ID resolution. Based on the Handle System.

DOI suffix – A unique identifier within a DOI namespace, which is combined with a DOI prefix to create a globally unique identifier. DOI suffixes are of different lengths and formats, depending on

the namespace. The EIDR Party suffix, for example, is 8 hexadecimal digits, while the Content suffix is 20 hexadecimal digits followed by an alphanumeric check character. In a DOI ID's canonical form, the DOI prefix is separated from the suffix by a slash ("/").

duplicate – An existing record determined to be identical to another record.

edit – Since "Edit" is a type of EIDR record, changes made to an existing record are referred to as "modifications" rather than "edits" to avoid confusion.

Edit – This content data type represents a content instance with unique characteristics that differentiate it from any other version. An Edit has different audio or visual content from that of its Parent.

EIDR – The Entertainment ID Registry develops and operates the EIDR System. EIDR is a DOI Registration Authority.

EIDR ID – Any of the several (currently four) types of IDs administered by EIDR. In general parlance, if the type of ID is not qualified (e.g., EIDR Video Service ID), then it is assume to be a Content ID ("EIDR ID" ≈ "EIDR Content ID"). In this document, we shall always make the distinction so "EDIR ID" shall reference EIDR IDs in general and only "EIDR Content ID" shall specifically reference Content IDs. EIDR is an implementation of the Digital Object Identifier (DOI) standard, so all EIDR IDs are DOIs.

EIDR System – A software system that includes the EIDR Registry and De-Duplication service. The software is used to create and maintain an object-oriented database of four sets of related records: Content, Parties, Users, and Video Services.

Episode – This content data type represents an episode of a Series. Can be a child object of an EIDR Series or Season object. An Episode must have one of the following values for its Referent Type: TV (most frequently), Movie, Web, Short, or Supplemental.

extra object metadata – A set of fields that supplement the base object data in certain EIDR record types, such as Compilations, Episodes, and Clips. Extra object metadata is never inherited to child records, though Series and Seasons may use extra object metadata flags to impose data validation constraints on the child records.

false negative match – Issuing a new EIDR ID to a duplicate record. This represents a failure of the de-duplication system, generally due to imperfect or incomplete descriptive metadata in either or both of the submitted record or the existing record; a flaw in the automated de-duplication algorithms; or human error in manual de-duplication. False negative matches may be corrected after the fact by aliasing one ID to another.

false positive match – Issuing an existing EIDR ID to an unrelated record. This represents a significant failure of the de-duplication system and should be avoided at all costs. Once released into the wild, there is no certain way of reversing a false positive match.

gap record – A record that is not currently in the registry.

Generic Manifestation – This Manifestation subtype represents the existence of a particular product, but does not include detailed technical metadata for the object. Its Structural Type is Digital.

greenfield data – A set of records that is not likely in the registry, such as records of a new Referent Type or local television content from a new market.

grouping record – A Compilation record or an Abstraction record with Composite details.

Handle System – A persistent identifier registry operated by CNRI (Corporation for National Research Initiatives), used as the basis for the DOI system. EIDR IDs are recorded in the Handle registry.

ID – There are four types of IDs in EIDR: User, Party, Video Service, and content. All EIDR IDs are also DOI IDs.

immediate mode – Synchronous processing mode.

Immediate Response – A mode for registering content records in EIDR. In this mode, all records are processed automatically without Manual De-Duplication in a synchronous or real-time manner. Users set this mode on or off for a request with a proprietary HTTP header.

In Development – One of two states for the EIDR Publication Status of a content record. This is used for records that need to remain generally hidden within the EIDR system until being promoted to "valid". Records in this state are not deduplicated.

inheritance relationship – An implicit relationship established between parent and child records, including: isSeasonOf, isEpisodeOf, isClipOf, isEditOf, and isManifestationOf.

Interactive Material – The Referent Type for assets that are not linear. It covers games, DVD menus, interactive TV overlays, customized players, etc.

IsAlternateContentFor – A Lightweight Relationship for alternate content that is synchronized to the specified record, such as audio or an alternate camera angle. (This should not be used for translations, which are usually handled by Manifestations.)

IsPackagingOf – A Lightweight Relationship for creating a collection of assets that are released together (such as a DVD).

IsPromotionOf – A Lightweight Relationship for promotional assets such as a trailer.

IsSupplementTo – A Lightweight Relationship for supplemental material that might be found on a DVD, such as an outtake or behind-the-scenes feature.

lightweight relationship – An explicit, non-inheritance relationship established between related records, including isAlternateContentFor, isPackagingOf, isPromotionOf, or IsSupplementTo.

Manifestation – This content data type represents an instance of a version that can be sold, transmitted, transferred or played. A Manifestation does not change the content, only the representation of it. This often includes additional language tracks. More specifically, a Manifestation object represents a particular instantiation of a Performance. See also Generic and Technical Manifestations.

manual de-duplication – Human review of match candidates when the automated process cannot make a high-confidence determination on its own. For synchronous transactions, users must perform their own manual review. For asynchronous transactions, EIDR Operations performs the manual review and the registry returns the final results to the user.

match candidate – One or more existing records that may duplicate the submitted record. Final determination requires manual review (by EIDR Operations for asynchronous transactions, by the submitting user for synchronous transactions).

Movie – In EIDR, this Referent Type represents content that first appeared in a theatre/cinema or was released directly to video. Note that this does not include a movie made for television broadcast. It is typically more than 40 minutes in length.

Namespace – The EIDR system contains four namespaces. One each for Users, Parties, Video Services and Content records. Each namespace has a unique DOI Name prefix.

no match – The submitted record is presumed to be new (no duplicates or match candidates were found). If part of a create transaction, the registry will then issue a new EIDR ID. If part of a match transaction, the match status is returned to the user.

non-immediate mode – Asynchronous processing mode.

parent – A record that has one or more child objects in the registry's inheritance tree. Child objects may inherit certain metadata from the parent.

Parent Object – A content record that has child objects. Parent and child are in an Inheritance Relationship.

Party – A corporate entity that is used in several ways: for associating Users with permissions; for access control on EIDR audiovisual records; for assigning roles in audiovisual records such as Registrant and Associated Organization.

Party ID – The EIDR ID for a company or organization record, identified by the DOI prefix 10.5237. Parties have defined roles in the EIDR security system that dictate how they may be used within the other registries (such as Associated Org or Metadata Authority within the Content ID registry) and how their Users may access the EIDR registries (such as Registrant or Alt ID Writer). An individual record's Access Control List (ACL) is assigned at the Party level.

Performance – A Structural Type that is used for a particular Edit or version of a work. It cannot be used for Series, Seasons, Episodes or Interactive Material.

A Record Type that includes Edits and Clips.

Provenance – In EIDR, this represents a group of system-defined data fields about a content record such as the Creation Date.

publication status – The value of a record's `Status` field (`valid, confidential, alias, restricted`). Most records are `valid`**;** `confidential` is used sparingly for confidential registrations; `alias` indicates an ID that was deprecated in favor of a surviving ID; and `restricted` is only used with the Tombstone record. When an "in development" record is promoted to "valid", it will be de-duplicated and if successful, it will be publicly visible.

Record – A record of any type in the EIDR Registry with an EIDR DOI ID.

Record Type – A record's essential nature. The Record Types are: Collection, Abstraction, Edit, and Manifestation. Collections include Series, Seasons, and Compilations. Abstractions include abstract root records and Episodes (including Composites).

Referent Type – This base field of a content record describes the abstract nature of the content's referent irrespective of its Structural Type. The Referent Type comes from an enumerated list: Compilation, Movie, Season, Series, Short, Supplemental, TV, Web.

Registrant – This base field represents the creator of the content record. This is the Party used in the credentials of the creation operation for the record.

Registration – The initial creation of a record in EIDR. When this operation succeeds, it creates an EIDR record and ID.

Registry – The component of the EIDR System that provides the EIDR HTTP API and stores the metadata for all records.

Relationship – A connection between content records in the EIDR System. See the three classes: Inheritance Relationship, Dependence Relationship, Lightweight Relationship.

resolution – A request to return the metadata for a given EIDR ID or Alternate ID.

resurrection – Repurposing a previously aliased ID, including IDs aliased to the Tombstone record (in lieu of deletion). A resurrected record can take on any valid form and is not limited by its prior state or original contents.

root record – The record at the top (or base, depending on your perspective) of an inheritance tree. This may be an Abstraction or Collection (Series or Compilation) record. For a Movie, this is the base object, for episodic television this is the Series record. The root of a tree can be any Referent Type except Season.

Season – A second level of grouping below a Series. Its child Episode records can have a Referent Type of TV, Movie, Web, Short, or Supplemental. Part of the Collection Record Type.

Series – In EIDR, a Series represents a video title that is divided into episodes. The episodes may or may not be subdivided into Seasons. The episodes most commonly have a Referent Type of TV, but may also be Movie, Web or Short. Part of the Collection Record Type.

Short – This Referent Type is used for content that is loosely defined to cover a work that is 40 minutes or less, such as music videos, theatrical newsreels, or theatrical or DTV cartoon shorts. (This is not meant for Web series episodes or any TV program in a regularly scheduled timeslot.)

status – A request to return the current state of a transaction via its Token ID. Valid for both system-assigned tokens (issued for every registry transaction) or user-defined tokens (associated with a transaction at the user's option).

Supplemental – This Referent Type is used for trailers, value-added materials, or miscellaneous content such as outtakes or special segments (such as documentary features).

synchronous mode – Records are processed within a defined timeout interval. If a timely response is not possible, then depending on the situation the registry will either return a timeout error or return a Token ID and convert the transaction to asynchronous processing.

Technical Manifestation – This Manifestation subtype gives details of containers, tracks, and card sets. Its Structural Type is Digital.

title record – A single record that describes a unique audiovisual work in the abstract; an EIDR Abstraction. **NOTE:** EIDR no longer uses the term "title record" and instead has re-defined "Abstraction" so there is a one-to-one correspondence between the terms. Still, the term "title record" may appear in other contexts.

tombstone record – An EIDR record that acts as the alias target for IDs that would otherwise have been deleted. (You cannot delete an EIDR ID once it is issued, so the next best thing is to alias it to the tombstone record.) The tombstone records cannot be used for any other purpose.

Token – A unique numeric ID assigned to each registry transaction. May be returned to the user as part of an asynchronous transaction. Can be queried with the API for status.

TV – This Referent Type is used for content that first appeared via broadcast. This includes one-time only (such as telefilms and specials) and episodes.

User – A type of record that is used to authenticate access to the EIDR System and then to track operations that write to the database. A User is a member of a Party.

User ID – The EIDR ID for an EIDR registry user's record, identified by the DOI prefix 10.5238. All Users are assigned to an EIDR Party.

Valid – One of the two most common states for the EIDR Publication Status of a content record. This is the typical value for released works.

Video Service – An audiovisual content provider, such as a broadcast network or channel, cable TV channel, satellite TV channel, video-on-demand (VOD) provider, etc., identified by a unique ID.

Video Service ID – The EIDR ID for an audiovisual content delivery channel record, identified by the DOI prefix 10.5239.

Web – This Referent Type is used for content that first appeared on the Web. This should not be used for content from elsewhere that has been made available on the Web.