Resolution API v1.03

THE TITLE REGISTRAR

Introduction
EIDR has the potential to be the keystone piece of the coming digital production and distribution infrastructure. With it, not only will workflow automations and vendor-to-vendor title data interchanges become possible, but new products technologies will emerge that depend on EIDR.

The Title Registrar's core mission is filling out the set of EIDR services and technologies that will allow genuinely universal content identification. An important part of this suite is our Resolution API. With it, anyone who depends on EIDR in their own business can:

- Verify that an EIDR ID exists
- Verify that an EIDR ID represents the correct kind of entity (e.g. performance, episode)
- Determine (and retrieve, if needed) an EIDR record's family (ancestor and descendent titles) to re-create a work's hierarchy for analytics or reporting.
- Update their own metadata from EIDR data
- Automate their own intake, workflow, and title reporting options
- Resolve EIDR IDs from 3rd-party IDs, and vice versa
- Search for, and match titles, based on limited metadata (Paid licenses only)

The Resolution API is simple in design, high-performance (>1,000 resolutions per second), and easy to access through formal integrations, or simple scripting in languages like Python or JavaScript.

API keys.
To start, you'll need to get an API key. The first step is to create Title Registrar account at titleManager.titleRegistrar.com, and then request a key. A one-month evaluation tier license (10,000 resolutions) can be obtained directly through a help desk request, but for higher-tier licenses, call us or open a Help Desk ticket to set up payment arrangements, and confirm acceptance of our API terms and conditions. They are pretty standard except for a few things:

1. Data obtained through the API may not be used for public-facing services or websites, only for your own company's internal use.
2. No one may use our APIs to create a service that competes with the Title Registrar.
3. Anyone trying to get clever to get around evaluation-tier licenses limits will be cut off entirely from API access (paid or otherwise). An example would be queries from two separate accounts coming from the same company, or the same IP address. If there are circumstances where this is natural and unavoidable, give us a call—we try to accommodate all reasonable requests.

If you fear that the security of your API key has been compromised, you can have a new one generated for you through our web application. This new key will have precisely the same limits, terms, and quotas as for the old key. The old key will be instantly invalidated, so once you generate a new one, you'll have to update your scripts and applications to use the new key.

Call metering.

Each API tier is limited to a certain number of "resolutions" every month (an API "month" runs from the first day to the last day of each calendar month), and in any single day (generally 1/5 of the monthly limit).

As a general rule, every title or party record returned through an API call "spends" one resolution credit. However, our API is not a jerk, so if you're running a call that would take you over the limit, it still provides you a complete result set.

If you make a call that returns no results, that *still* consumes one resolution credit. AltID calls each consume one credit, as well, regardless of how many records it might return. Title matching and searching are not available on the evaluation tier, and will cost 5 extra credits per call, as they are heavyweight and resource-intensive.

There is a "quota" call that tells you how many of your credits you've consumed and how many you have left for the day and month. This call is "free," and is not counted towards your license limit.

Title Summary records

Most API calls return "summary" records of the EIDR information, which includes the most important data from the title record, plus contextual information from the title's family (the parent, patriarch, and title-level title), and provenance information (registrant, version number, and creation / update dates).

The summary record includes the following:

Type	Name	Description
string	**eidrID**	The EIDR identifier of the record
string	**title**	The primary title of the referenced work
enum	**titleType**	The "type" of work represented (e.g. Movie Edit). See Appendix
short	**releaseYear**	Year of first commercial release
string	**directors**	Directors of work
string array	**cast**	Cast members (up to four)
string	**runtime**	Runtime of work, in {#h#m#s form) e.g. (2h03m). In the Python helper library, there is a function to convert this to minutes, if needed
string	**parentEidrID**	EIDR ID of parent record of the current one (if any)
string	**patriarchEidrID**	EIDR ID of the patriarch record of the current record's family
string	**titleLevelEidrID**	EIDR ID of the closest abstract ancestor (from which the current record derives its title)
unsigned short	**version**	Revision number of the current EIDR record (which goes up by 1 every time the EIDR record is revised)
string	**registrantID**	EIDR party ID of the entity that first registered this record
string	**registrantName**	The primary *name* of the registering party
datetime	**createdt**	Timestamp of initial registration date (GMT)
datetime	**lastUpdDt**	Timestamp of last revision to EIDR record (GMT)
string	**sorter**	(included only in multi-record family calls[1]
Object	**registration**	If this EIDR record was created for your company through the TTR, registration information is included (see Appendix B)

[1] A sorting key that can be used to order a result set in "tree" order (recursive descent). When there are siblings, the sorting key will attempt to order them in enumerated order (e.g. for seasons and episodes) by primary distribution order; if unable to do that, it will order the records by release date; and if that doesn't resolve things, alphabetical title order is used

Full EIDR records (in either XML or JSON representation) can be obtained through a “full” resolution call

Summary records are used for most operations because they are short, have all the metadata that most workflow operations need, and are consistent in structure and easy to work with.   They include hierarchy and provenance data that might consume three to ten additional calls on EIDR’s API to follow ancestry chains.

API Calls

There are about a dozen different API calls (or endpoints) for getting EIDR data. Most are available on all licensing tiers, but there are a couple of exceptions which are not available through our evaluation tier. These will be noted below.

API key. Access to the API requires that an API key be included in the HTTP header of each request, with the key "apikey." Any calls made using your API key will be charged to your account, so only share keys if you're willing to live with the consequences.

Calls. Virtually all API calls are GETs. *Search* and *Match* calls are POSTs.

EIDR IDs

When an API call accepts an EIDR content ID, or Party ID, you may include the entire string, or only the part after the prefix (i.e. starting after the slash). We have found that many IT shops store IDs without the prefix, and the embedded slash in EIDR record IDs can be confusing given standard API request nomenclature, which uses slashes to separate parameters in a call. So, to the API, all of the following ID strings are seen as functionally identical:

- `10.5240/5868-409E-7BFB-536A-6067-E`
- `5868-409E-7BFB-536A-6067-E`
- `5868-409E-7BFB-536A-6067`
- `5868409E7BFB536A6067E`
- `5868409E7BFB536A6067`

This goes for both title and party IDs

Single-record resolutions

These calls query for a single title record, and return it—if found—in different forms.

full/{EidrId}	GET	Returns a full (JSON) title record, which is more comprehensive, but harder to work with than the TitleSummary.
xml/{EidrId}	GET	Returns a traditional XML version of the EIDR record, which may be useful if you're in an IT shop that already has an existing EIDR integration using EIDR's REST API.
title/{EidrId}	GET	Returns a single TitleSummary record
title/empty	GET	Returns a single, empty, TitleSummary record. There are a couple POST calls that require populated TitleSummary data in their body, so this is an easy way to get a starting point.

Most of the title resolution calls (except for **full** and **xml**) return TitleSummary records, which provide a compact and easy-to-use return object format that combines several different record types from the registry into a single record which includes primary metadata, provenance data, and hierarchy pointers for building family or roll-up hierarchies. A TitleSummary response looks like this:

```
{
    "EidrID": "10.5240/F562-5D87-0371-87A2-F66E-T",
    "Title": "Star Wars",
    "TitleType": "Movie Edit",
    "ReleaseYear": 1977,
    "Directors": [
        "George Lucas"
    ],
    "Cast": [
        "Mark Hamill",
        "Harrison Ford",
        "Carrie Fisher",
        "Alec Guinness"
    ],
    "RunTime": "2h4m",
    "ParentEidrID": "10.5240/5868-409E-7BFB-536A-6067-E",
    "PatriarchEidrID": "10.5240/5868-409E-7BFB-536A-6067-E",
    "TitleLevelEidrID": "10.5240/5868-409E-7BFB-536A-6067-E",
    "Version": 1,
    "RegistrantID": "10.5237/2FE2-24F2",
    "RegistrantName": "20th Century Fox Film Corporation",
    "CreateDt": "2019-07-16T16:55:14Z",
    "LastUpdDt": "2019-10-30T20:56:31Z"
}
```

Title hierarchy resolutions

Much of the power of EIDR—and its future—comes from the relationships between members of a record's "family." This could include performance-level records, Series→Season→Episode→Edit→Manifestation hierarchies, etc. These can be fundamental in doing royalty and play reporting, analytics, and the like.

In fact, working with EIDR hierarchies is kind of "baked-in" to the approach we have taken in this API. The TitleSummary record not only has the key fields for each record (title, runtime, cast, version, etc.), but also the reference EIDR IDs of related records: parent, patriarch (top-level title of any title family), and title-level title (the abstract record which heads the hierarchy within a family for a single work).

The following calls allow you to retrieve title records related to a particular title of interest:

Ancestors. EIDR records from which a given title was derived (i.e. the titles above it in the hierarchy).

Descendents. EIDR records which are derived from a given title (i.e. the titles *below* it in the hierarchy).

Peers. EIDR records which share the same parent, as with episodes in a season, or edits of a movie.

Family. All the titles that derive from the same patriarch as the given title. This returns the entire family.

The Sorter field.

All of these hierarchy resolution calls populate a field in the TitleSummary record called "Sorter". If returned records are ordered by the Sorter field, the records will be organized in recursive descent order. They are already ordered this way in the API call results. We try to make the sorter field work as described here, but some EIDR records don't allow a "natural order" to be safely inferred: e.g. episodes without sequence numbering, and the same date.

We do our best. Please don't yell at us.

ancestors/{EidrId}	GET	Returns all of the records from which the given title derives, all the way up to the patriarch. Note that this includes *only* the titles in the title's direct lineage (father, grandfathers, great-grandfathers) but does not include uncles or cousins.
descendents/{EidrId}	GET	This is the complement of the *ancestors* call, and includes all titles that have the given title in its lineage. A Season record, for instance, might have dozens of Episodes, which in turn have more dozens of Edits.
peers/{EidrId}	GET	Returns all EIDR records that share the same parent as the given record. For a Season Episode record, for example, this call would return the *other* episodes from that season.
family/{EidrID}	GET	Returns all titles related to the given record ID, from the patriarch down to the lowest leaf level.

Search and Match

These calls allow searching for EIDR records by a number of different criteria. A couple of them are quite resource intensive, or make direct calls into EIDR's API, so are not available in the evaluation API tier (and cost extra credits).

match POST

(Paid tiers only, 5 extra call credits). The body of this request is a TitleSummary record, populated with as much metadata as is currently known (e.g. Title, release date, runtime, director(s), cast members, etc.). From this, the API will create a Match call to EIDR, then "curate" the result set to respect any restrictions placed in the submitted record (e.g. registrant name, family EIDR IDs), or obviously non-matching records.

This can be helpful to help determine whether your catalog titles already have EIDR IDs, or if an incoming title from a vendor has an ID or needs to be registered. (If it *does* need to be registered, remember your friends at *The Title Registrar*.)

These will not be "perfect" matches, and will not be as accurate as matching against a record which has full metadata. But, since this call has the potential to save your organization a lot of wasted time and resources, we've included it as imperfect as it is.

An example **match** response is included in Appendix A.

> The *LowMatchThreshhold* is the minimum match score that EIDR considers to be considered for that match request to be *candidate* matches for the provided information.
>
> The *PositiveMatchThreshhold* is the score value at which EIDR considers the title to be a high-confidence match for the submitted data.
>
> *MaximumScore* is the highest score returned among *all* of the match candidates returned by the call. Also, the individual match candidates are ordered by their match score, so the closest matches are at the top of the list.
>
> *HasPositiveMatch* is set to ***true*** if any of the candidate titles meet EIDR's *PositiveMatchThreshhold* for a high-confidence match
>
> *MatchCount* is simply the number of identified match candidates for the submitted title metadata.

find POST

(Paid tiers only, 5 extra call credits) The body of this request is a TitleSummary record filled out with as much known (accurate) data as you're able to provide.

We have a fancy new back end to support these searches, which are score-based fuzzy matching, and are pretty performant. Search results are limited to no more than the 20 highest-scoring candidates, and are ranked in descending score order (i.e. with the most likely match candidates displayed at the top of the list).

If the "title type" includes a record type, or referent type, the candidate list will be pruned to conform to it (so if you're looking for root titles, include "Basic" in the titleType field, for Edits include "Edit", etc." -- it's used as a filtering criterion, so it doesn't have to be a precise match for a full title type).

.

altid/ {idType}/ {idValue}/ {domain?} GET

Returns all EIDR records that match the given Alternate ID. This can be a handy way to match a title from an IMDB ID, or an internal catalog number. The parameters are as follows:

idType. This must be one of the EIDR designated ID types (case-sensitive): ISAN, TVG, AMG, IMDB, Baseline, MUZE, TRIB (Tribune), UUID, URI, Grid, ISRC, DOI, SMPTE-UMID, AD-ID, UPPC, CRID, cIDF, IVA, or URN.

All other ID types must be categorized as “Proprietary”, and use a distinguishing domain name to define them. Some common Alternate ID types that don’t have dedicated EIDR types are:

ID type	**Domain**
Flixster	flixster.com
Netflix	netflix.com
CinemaSource	thecinemasource.com
Amazon ASIN	amazon.com
CITWF	citwf.com

Note that domain names are always specified in all lower case.

If you’re unsure of the code or domain for a given ID, you have the option of specifying “any” as the ID type, in which case the search will return AltID summaries for all titles with the given ID.

idValue. This is the Alt ID you want to match against. Matches are exact. Searches for the ID values “See-TiVO” and “Kukla.tv” are specifically excluded, since they are not real identifiers, and would return thousands of records.

domain. *(Optional)* This may be included *only* if the idType is “Proprietary,” and represents the distinguishing domain for a Proprietary title. Note that the domain should be all lower-case, and should *not* include “http://”, “https://”, or “www.” prefixes.

Alternate IDs

One of the powers of EIDR IDs is that it eases the exchange of data between vendors, partners, departments, customers, etc. Someday, everyone will be using EIDR numbers to do this. But today, many parties have their own proprietary IDs, or use IMDB or Flixster IDs or something similar to identify their titles.

Once again, EIDR comes to the rescue. Any Alternate ID of the same scope may be added to an EIDR record, which allows "mapping" from an external ID, to an EIDR ID, and perhaps to a *different* external ID. In that way, EIDR not only helps with cross-company (and cross-department) title data integrations, but has provided a way to do transitional matching until the time that everyone uses EIDR as their primary identifier.

For a popular work like *Star Wars*, there can many Alternate IDs mapped to the title—something like 43!

Full EIDR records have all alternate IDs included. But if you need to just extract a single alternate ID (or _all_ alternate IDs) for a particular EIDR record, there is a call to do that:

getAltIds/ {EidrId}/ {idType}/ {domain?}	GET	Returns AltID record(s) for the given EIDR ID. If the *IdType* is specified, the results will be constrained to AltID mappings of that type (or, if Proprietary, to that type and domain). If the idType is specified as "any," any AltID mapping with that specified ID value will be returned, regardless of type.
		.

getAltIds/ {EidrId}/ {idType}/ {domain?} Returns all titles that match the altID criteria specified. This call is described in the "Search and Match" section above.

Alternate ID records look like this:

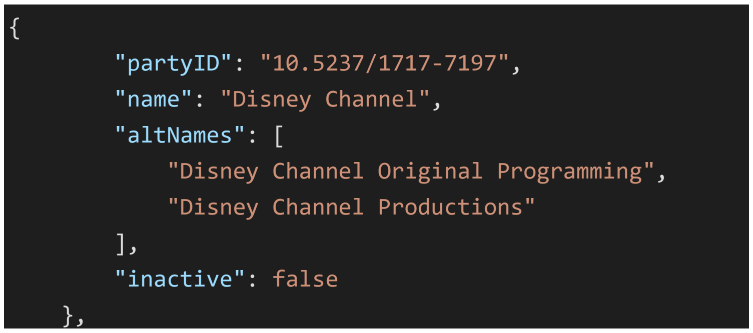
```
[
    {
        "altIdType": "ISAN",
        "altIdValue": "1255440",
        "altIdDomain": "commonsense.org/nid",
        "altIdToEidrRelationship": "IsSameAs",
        "eidrID": "10.5240/5868-409E-7BFB-536A-6067-E",
        "primaryTitle": "Star Wars",
        "releaseYear": 1977,
        "eidrType": "Movie",
        "recordType": "Basic"
    },
    {
        "altIdType": "ISAN",
        "altIdValue": "a119d85b-674b-4e95-a735-45ddf40fd661",
        "altIdDomain": "commonsense.org/uuid",
        "altIdToEidrRelationship": "Unspecified",
        "eidrID": "10.5240/5868-409E-7BFB-536A-6067-E",
        "primaryTitle": "Star Wars",
        "releaseYear": 1977,
        "eidrType": "Movie",
        "recordType": "Basic"
    }
]
```

EIDR Parties

EIDR records general refer to other parties in the EIDRverse by their EIDR party IDs—this goes for registrants, Associated Organizations, and access lists.   Here are a couple calls that will help you work with these

party/{PartyId}	GET	Resolves an EIDR Party ID to a Party record, described below
party/find/{words}}	GET	If you *need* and EIDR Party ID, this can help you resolve in the other direction. The *words* parameter should include words (or word fragments) expected to be in a party's name (including alternate names). For example, you might use "Warner", or "Disney Animation". Word fragments are okay – this call is just looking for substrings in one of a Party's names. Results are returned by their similarity to the provided word string, with the strongest matches at the top of the list.

An EIDR Party record looks something like this:



```
{
        "partyID": "10.5237/1717-7197",
        "name": "Disney Channel",
        "altNames": [
            "Disney Channel Original Programming",
            "Disney Channel Productions"
        ],
        "inactive": false
    },
```

Account Management

Since this is a metered API, you probably don't want to be taken by surprise and get cut off in the middle of the month for exceeding your API call quota. We've provided a call that will tell you about the account status associated with a single API key – what your daily and monthly usage quotas are, and how many calls you still have left. This call is "free," and doesn't burn any API credits.

quota GET

Returns the API account status, including
How many calls per day/month you're entitled to
How many you've made so far
How many you have left.

A quota response looks something like this:

```
{
    "dailyMax": 2000000,
    "monthlyMax": 10000000,
    "callsToday": 27307,
    "callsThisMonth": 27307,
    "callsLeftToday": 1972693,
    "callsLeftThisMonth": 9972693
}
```

.

changed/{sinceDt} GET

For an EIDR member, there may be a desire to maintain internal systems to remain in sync with EIDR records, or to review changes to EIDR records of their titles done by other parties (or even by other departments. This call supports these activities by returning EIDR records that are new, or have been changed, since the hallmark date provided.

In order to use this function, you will have had to register your EIDR Party ID with us, so we can filter the new titles returned to include only them. Currently, this must be done through a help desk ticket, although our new website will allow it to be managed there.

If you are a Title Registrar registration customer, this call will return the titles registered by you through our agency. This requires no particular additional action on your part.

This call will return no more than 200 records per call—if you receive a "Quota Exceeded" response, you must narrow the date range so that the call can complete.

Batch Management

Most customers primary use for the API is to maintain internal catalogs with EIDR information, and to keep it up to date. Towards this end, there are a number of calls to assist:

reg/pymtid/*{pymtID}*	GET	Returns title summary records (with registration information) for all titles associated with a particular payment or invoice ID.
reg/since/*{daysback}*	GET	Returns title summary records (with registration information) for all titles registered (or updated) by your company through TTR in the last *{daysback} days.*
reg/pending	GET	Returns title summary records (with registration information) for all pending TTR registrations for your company.
reg/complete/*{daysBack}*	GET	Returns title summary records (with registration information) for all titles registered through TTR on behalf of your company in the last *{daysback}* days
get/patriarch/*{eidrID}*	GET	Returns patriarch ID of the given title, as an EIDR ID string
recent/ids/*{daysBack}*	GET	Returns a list of EIDR IDs for TTR registrations on behalf of your company (even those still in processing}. If a title has not yet received an EIDR ID, the ID will be a string "rs*RtID",* where RtID is the TTR Registration Transaction ID
recent/patriarchs/{daysBack}	GET	Returns a distinct list of the patriarch EIDR IDs of all TTR-registered titles for your company created in the last *{daysBack}* days.

Appendix A

Sample match response

```
{
    "LowMatchThreshhold": 60,
    "PositiveMatchThreshhold": 95,
    "MaximumScore": 87,
    "HasPositiveMatch": false,
    "MatchCount": 50,
    "Matches": [
        {
            "MatchScore": 87,
            "EidrID": "10.5240/04E7-58E4-EE8D-F1EC-F4DD-G",
            "Title": "Love",
            "TitleType": "Movie",
            "ReleaseYear": 1982,
            "Directors": [
                "Liv Ullmann",
                "Mai Zetterling"
            ],
            "Cast": [
                "Joni Mitchell",
                "Dixie Seatle",
                "Margaret Dragu",
                "Nicholas Campbell"
            ],
            "RunTime": "1h45m",
            "ParentEidrID": "",
            "PatriarchEidrID": "10.5240/04E7-58E4-EE8D-F1EC-F4DD-G",
            "TitleLevelEidrID": "10.5240/04E7-58E4-EE8D-F1EC-F4DD-G",
            "Version": 1,
            "RegistrantID": "10.5237/A0CA-AA66",
            "RegistrantName": "Internet Video Archive",
            "CreateDt": "2020-03-31T19:28:02Z",
            "LastUpdDt": "2020-03-31T19:28:02Z"
        },
        {
            "MatchScore": 85,
            "EidrID": "10.5240/01BE-D75A-0676-620B-83E3-P",
            "Title": "Izakaya Choji",
            "TitleType": "Movie",
            "ReleaseYear": 1983,
            "Directors": [
                "Yasuo Furuhata"
            ],
            "Cast": [
                "Ken Takakura",
                "Reiko Ohara",
                "Tokiko Kato",
                "Kunie Tanaka"
            ],
            "RunTime": "2h6m",
            "ParentEidrID": "",
            "PatriarchEidrID": "10.5240/01BE-D75A-0676-620B-83E3-P",
            "TitleLevelEidrID": "10.5240/01BE-D75A-0676-620B-83E3-P",
            "Version": 1,
            "RegistrantID": "10.5237/F625-DC51",
            "RegistrantName": "The Title Registrar",
            "CreateDt": "2017-12-18T00:15:25Z",
            "LastUpdDt": "2019-01-26T22:37:00Z"
        },
        {
            "MatchScore": 82,
            "EidrID": "10.5240/F518-F443-43A2-195D-3EC4-Y",
            "Title": "Ljubezen",
            "TitleType": "Movie",
            "ReleaseYear": 1985,
            "Directors": [
                "Rajko Ranfl"
            ],
            "Cast": [
                "Rok Bogataj",
                "Lenka Ferencak",
                "Bernarda Gaspercic",
                "Vesna Jevnikar"
            ],
            "RunTime": "1h",
            "ParentEidrID": "",
            "PatriarchEidrID": "10.5240/F518-F443-43A2-195D-3EC4-Y",
            "TitleLevelEidrID": "10.5240/F518-F443-43A2-195D-3EC4-Y",
            "Version": 1,
            "RegistrantID": "10.5237/F625-DC51",
            "RegistrantName": "The Title Registrar",
            "CreateDt": "2017-12-18T14:59:58Z",
            "LastUpdDt": "2017-12-30T22:17:35Z"
        }
]
```

Appendix B

Sample registration information record

When a registration summary record refers to an EIDR record that has been created through the Title Registrar on behalf of your company, the ***registration*** attribute includes registration information for that title, including the employee name that performed the registration, the charge, and the current status of the registration record.

A sample registration record is included below

```
"Registration": {
           "RtID": 101311,
           "TransactionType": "Create",
           "Status": "Complete",
           "CustID": 29227,
           "CustName": "Example Distribution, Inc.",
           "Registrant": "Theodore Sample",
           "Charge": 21.0,
           "PaymentID": 123456,
           "SubmitDt": "2021-10-11T19:26:03.847",
           "StatusDt": "2022-03-13T15:37:24.647"
       }
```