EIDR SYSTEM
VERSION 2.1

REST API REFERENCE

2016 Jan. 4

EIDR REST API Reference.

Feedback on this document can be sent to support@eidr.org

TABLE OF CONTENTS

1 INTRODUCTION 4

1.1 Public Services 4

1.2 DOI Proxy 6

2 HTTP AND REST API ELEMENTS 7

2.1 Common HTTP Headers 7

2.1.1 HTTP Request Headers 7
2.1.2 Authentication and Authorization 8
2.1.3 POST Data 8
2.1.4 Codes and Descriptions 9
2.1.5 HTTP Response Headers 13

2.2 XML Schemas 13

2.3 Public Services API 13

2.3.1 Registration service 14
2.3.2 Status Lookup service 18
2.3.3 Resolution service 21
2.3.4 Match Service 26
2.3.5 Query service 29
2.3.6 Graph traversal service 32
2.3.7 Modification base service 34
2.3.8 Virtual Fields Retrieval service 36
2.3.9 Party Resolution Service 37
2.3.10 Party Query Service 39

1 INTRODUCTION

The EIDR system provides various services as summarized in the ***EIDR Registry User's Guide*** using a REST-based interface in combination with HTTP 1.1 (see RFC 2616). Note that public services do not necessarily mean open access. Ingesting or registering data into EIDR is controlled, while reading data from EIDR is generally not restricted.

1.1 PUBLIC SERVICES

EIDR provides the following public services:

- **Registration service:** This service allows authorized users to perform the following content operations:
    - Create Object,
    - Add Relationship,
    - Remove Relationship,
    - Replace Relationship,
    - Modify,
    - Delete,
    - Alias,
    - Promote.

  For complete details about each of the operations listed, refer to the ***EIDR Registry User's Guide***. Note that the service allows batching identical operations on unrelated objects as part of the same request. The service returns a token for clients to check on the status of the batch. The service may also return the status as a response in order to support synchronous registrations (which have an immediate pass/fail flag). In the case of immediate pass/fail, only one operation must be part of the batch.

- Status Lookup service: This service allows authorized users to get the status of valid content write requests that were made previously. This service accepts the following types of request:
    - Token. A token may refer to a submitted batch or a single operation (within a batch). Please refer to the EIDR schema for the various kinds of status responses.
    - User ID. The response would be a list of tokens and their status. The list includes all requests made by the user. Optionally filters may be added to this request based on either status, or range of submission timestamps, or after-timestamp to return status of requests submitted later than the one specified.
    - Registrant**.** The response would be a list of tokens and their status. The list includes all requests made by users of this Registrant. Optionally filters may be added to this request based on either status, or range of submission timestamps, or a timestamp to return the status of requests submitted later than the one specified.

  For each of the status responses, a description may also be returned to provide more guidance to the users, especially when the request resulted in a failure. Specifically,

    - When the status type is a "duplicate", the response also includes one or more content ID's of previously registered objects. If there is only one match and that match is considered a perfect match, the ID will also be shown in the Status element.
    - When the status type is "success", the response also includes the content ID of the newly registered object.

The response is paginated. For operation requests with immediate response flag set to true, the status is available synchronously.

- **Resolution Service**: This service allows anyone to resolve a DOI to its metadata and related information. Filters may be specified to allow or disallow following alias chains. Depending on the type of resolution request, for each valid DOI, the response would be one of DOI kernel metadata, simple metadata, full metadata, self-defined metadata, or provenance. For objects which are aliased and following alias chain is not tractable, the last DOI in the alias chain that is tracked so far is returned.

  Depending on the access privileges of a user, some or all of those possible resolutions are restricted.

- **Query Service:** This service allows authenticated users to submit a query on registered metadata records and get a response. The response would be a list of records that matched the requested criteria. The service processes the request taking the access privileges into consideration and only returns those records the user has "read" access to. For complete details on how the results are screened based on the access privilege, please refer to the ***EIDR Registry User's Guide*** documents. The response is paginated.

- **Graph Traversal Service**: This service allows authenticated users to request certain aspects of object relationships. The following types of sub-graphs may be requested:

  - Ancestors of an object, which may be filtered for specific referent types, structural types, and/or relationship types.
  - Descendants of an object, which may be filtered for specific referent types, structural types, and/or relationship types.
  - Series ancestry.
  - Remotest ancestor of an object.
  - Leaf descendants of an object.
  - Parent of an object.
  - Children of an object.

  Note that in the traversal if the registry encounters an object with restrictive reads (that is, objects with a status of "in development"), the registry continues its traversal only if the user is authorized to read that object. Refer to schema for return values.

- **Modification Base Service**: This service allows authenticated users to retrieve the XML required to create the object as a specified type. The returned information can be used to produce appropriate input for the Modify operation. See ***EIDR Registry Programmer's Guide*** for details.

- **Virtual Fields Retrieval Service**: This service retrieves the values stored for each of the virtual fields for the specified object. Refer to the ***EIDR Registry User's Guide*** for the list of virtual fields. Each of the virtual fields is a combination of multiple fields from the specified metadata for an object. Those virtual fields make it easy to target a combination of fields using the Query service. This helper service allows retrieving the actual strings stored for the virtual fields and would be useful for debugging reasons to verify why or why not an object is returned when a developer or user thinks it should be otherwise.

1.2 DOI PROXY

See the ***EIDR Registry User's Guide*** for a description of the DOI proxy's behavior and parameters for EIDR resolutions.

2 HTTP AND REST API ELEMENTS

This section discusses the API of the various services offered by the EIDR.

The production registry is accessible only through the HTTPS protocol.

2.1 COMMON HTTP HEADERS

As is the case with all REST implementations using HTTP, all requests and responses vary in the HTTP method used for requests, HTTP headers, and the actual data transmitted between the server and the clients. The following three sub-sections summarize the headers used for all the EIDR requests and responses.

2.1.1 HTTP REQUEST HEADERS

The EIDR API uses the following HTTP request headers for all service requests.

Header Name	Required	Description
Accept	Optional	The MIME type of the accepted response. Defaults to "`text/xml`". The value should be `text/xml`.
Accept-Encoding	Optional	Enables HTTP compression. If not present, then defaults to no compression. If specified, the value should be `gzip`. If the Registry uses compression, then the Content-Encoding response header will confirm.
Authorization	Conditional. Required for only access-controlled service requests. See Section "Public Services" to find which ones are access-controlled and which are not.	The required authentication and authorization credentials. See the "Authentication and Authorization" section below for the value.
Immediate-Response	Optional, but conditional. Refer to the ***EIDR Registry User's Guide*** to know which operations can be "`true`" and which cannot.	This is a proprietary header that applies only to registration requests. It is a flag that if true indicates that the service should process the request immediately. The default is "`false`".
Content-Type	Required for the POST method	The value can be `text/xml` or `multipart/form-data` in EIDR requests. Text/xml is recommended since it is simpler and the request can be validated with an XML validator.

EIDR-Version	The version of the EIDR REST API for the request	Setting this to a value less than the current version will return a response compatible with the request. If this is not possible it will be return a 23 code (compatibility error).

Table 1: HTTP Request Headers

Note that the EIDR ignores any other request headers if specified in a request.

2.1.2 AUTHENTICATION AND AUTHORIZATION

The EIDR API uses the standard HTTP Authorization header to pass both the authentication and authorization credentials. The EIDR API uses a proprietary authentication scheme ("Eidr") that extends standard HTTP Basic authentication to incorporate both an EIDR user and an EIDR party into the credentials. A pseudo-grammar of the HTTP Authorization request header is shown below:

```
Authorization = "Eidr" + " " + UserID + ":" + PartyID + ":" + PasswordShadow;
PasswordShadow = Base64(MD5(Password));
```

Note the "Eidr" authentication scheme token is case-insensitive.

Also note that as required in Base64 encoding, the resulting string must include padding out to a multiple of 4 characters by appending "=" characters. In the case of a 128-bit MD5 hash, the base64 encoding results in a 24-character string ending in "==". Some APIs, such as Perl's md5_base64() function, leave the addition of padding to the caller. Also note that the base64 encoding of the MD5 hash is of the binary string, not of the hexadecimal text string.

2.1.3 POST DATA

Note that certain characters commonly used in the XML elements need to be escaped. Most XML libraries perform this escaping automatically. The XML characters that require escaping are listed in the table below.

Character glyph (name)	XML Escape
" (double quote)	`&quot;`
' (apostrophe, back quote)	`&apos;`
< (less than)	`<`
> (greater than)	`>`
& (ampersand)	`&`

Table 2: XML character escaping

For POST requests, the data can use the multipart/form-data MIME type (see IETF RFC 2388).

In the multipart case, the POST data must begin with a boundary. The initial boundary is followed by these headers:

Header Name	Required	Description
`Content-Disposition`	Required	Where the value of the name parameter usually matches the salient part of the path of the URI for the service. For example `name="query"` for the `/EIDR/query` service.

`Content-Transfer-Encoding`	Required	This is always binary: `Content-Transfer-Encoding: binary`

Table 3: HTTP Multi-part MIME Headers

Note that these lines must be terminated CRLF (per RFC 882 and 2045). The final header and the body must be terminated by two CRLF sequences. The part is terminated by another boundary. A sample POST file for a query (where the boundary is "---313159"):

```
---314159
Content-Disposition: form-data; name=query
Content-Transfer-Encoding: binary


<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.eidr.org/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Operation>
  <Query>
   <Expression>/FullMetadata/BaseObjectData/Status valid</Expression>
   <PageNumber>1</PageNumber><PageSize>25</PageSize>
  </Query>
 </Operation>
</Request>

---314159--
```

2.1.4 CODES AND DESCRIPTIONS

EIDR will always respond with an HTTP Status Code of "`200 OK`". If there is an error, it will be in the response body. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0">
  <Status><Code>8</Code><Type>bad id error</Type></Status>
</Response>
```

The set of applicable Status Codes to the API request is as follows:

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.

Status Code	Status Type	Note
3	invalid request	An API (URI) that does not exist including missing a required parameter. May also include an incorrect HTTP operation on a valid URI (such as a GET on a registration). Could also be POST multipart 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 REST 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
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

Status Code	Status Type	Note
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	An Administration API error.
21	duplicate service	An Administration API error.
22	bad service	Invalid Video Service ID.
23	compatibility error	The operation cannot be supported with the requested value in the EIDR-Version header.

Table 4: API Status Codes

When the Request is a content Operation, the Response also includes an Operation Status code and will usually include a Details field. For example:

```
<RequestStatusResults>
…
 <OperationStatus>
  <Token>1312315566343000884</Token>
  <Status>
   <Code>4</Code>
   <Type>validation error</Type>
   <Details>Referent Type must be one of "TV", "Movie", "Short",
"Web"</Details>
  </Status>
 </OperationStatus>
</RequestStatusResults>
```

The set of Operation Status Codes is as follows:

Status Code	Status Type	Note
0	success	The data operation Request succeeded.
1	duplicate	Request failed because the system identified an existing record with duplicate metadata. One or more Duplicate ID elements will accompany this status.

Status Code	Status Type	Note
2	pending	This status normally occurs only in asynchronous processing. The state applies initially while the request is being processed, which might take several seconds. If it takes longer than that then the request is pending because the system identified an existing record that is a *potential* duplicate, in which case the registration will be manually evaluated. In all cases, this token should be polled until its status is “success” or “duplicate” or “rejected”.
3	authorization error	Your credentials are not authorized to perform this operation. Check with EIDR support about this operation.
4	validation error	The request failed because it did not satisfy data validation rules. For example, an attempt was made to modify a record to have an incompatible Referent Type (such as convert a Movie to a Series).
5	other error	Request failed due to uncategorized error. Contact EIDR support for details.
6	rejected	A request in manual deduplication failed due to a problem with the record. This is not common and you may wish to contact EIDR support about such a result.

Table 5: Operation Status Codes

All Operations statuses are terminal except for “pending”.

When the request consists of a batch with multiple operations, a Batch status applies. The set of Batch Status Codes is as follows:

Status Code	Status Type	Note
1	batch received	The batch has been fully read in, but no status for its individual operations is available. This is the state in the initial registration response. It is not a terminal state, but is transient and is usually only seen when the system is under heavy load. The token can be polled until its state reaches terminal status.
2	batch queued	Status is available for all the operations tokens. While this state is terminal for a batch, it does not mean that all tokens are in a terminal state. Specifically, a token could be “pending”.
3	invalid batch	A terminal status for batches that begin processing but terminate in an error before being parsed into individual operations. This includes batches with invalid user tokens. This can also result from abnormal operation of the Registry, which should be reported to EIDR support.

Table 6: Batch Status Codes

Refer to the ***EIDR Registry User’s Guide*** for more detail on the various codes and their descriptions.

2.1.5 HTTP RESPONSE HEADERS

The EIDR API responds with the following headers:

Header Name	Description
Server	This will always be: `Apache-Coyote/1.1`
Content-Type	The Internet Media (MIME) Type of the response sent by EIDR to its clients which is always: `text/xml; charset=UTF-8`
Content-Encoding	The compression applied to the content if any. If applied, this will only be: `gzip`
Content-Length	The length of the response in bytes. If not present, then Transfer-Encoding applies.
Transfer-Encoding	The compression applied to the data transmitted if applicable (as in gzip Content-Encoding). If applied, this will only be: `chunked`
Date	The time of the response. For example: `Mon, 08 Dec 2014 23:00:38 GMT`
EIDR-Version	This will always be the current version. For example: 2.0.7

Table 7: HTTP Response Headers

2.2 XML SCHEMAS

The most important schemas are:

- common.xsd – basic structures for Assets, Parties, and Users
- api.xsd – XML for formatting requests and receiving responses
- api-common.xsd – enumerations of error codes and strings.
- md-v21-eidr.xsd - EIDR additions to MovieLabs Common Metadata.

2.3 PUBLIC SERVICES API

This section defines the actual HTTP headers, parameters, and data transmitted for each of the services.

All services except resolutions and virtual fields retrievals are supported using HTTP POST. Resolutions and virtual fields retrievals are supported using HTTP GET only. Simple Status checks are supported using HTTP GET, while more complex queries require HTTP POST.

2.3.1 REGISTRATION SERVICE

This service is used for all read and write operations to the content database.

Name	Value	
URL	https://<*host*>.eidr.org/EIDR/register/	
Method	HTTP POST	
Encoding	`text/xml` or `multipart/form-data`	
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".
Immediate-Response		See section "HTTP Request Headers".
HTTP POST Parameters		
Name	**Type**	**Notes**
batch	XML	Refer to schema for XML.

Table 8: Registration Service Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type		See section "HTTP Response Headers"
Data		
Response	XML	Refer to schema for XML.

Table 9: Registration Service Response

The format of a registration request using multipart/form-data is as follows:

```
POST https://<host>.eidr.org/EIDR/register/ HTTP/1.1
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
Immediate-Response: true
Content-Type: multipart/form-data; boundary=---314159
---314159
Content-Disposition: form-data; name=batch
Content-Transfer-Encoding: binary
<Request XML goes here>
---314159--
```

The format of the response to a registration request is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8


<Response XML goes here>
```

Sample XML of a registration request to Create (an Episode in this case) is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.eidr.org/schema"
xmlns:doi="http://www.doi.org/2010/DOISchema"
xmlns:md="http://www.movielabs.com/schema/md/v2.1/md "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Operation>
  <Create type="CreateEpisode">
   <Episode>
    <BaseObjectData>
     <StructuralType>Abstraction</StructuralType>
     <AssociatedOrg idType="EIDRPartyID" organizationID="10.5237/62B6-3532"
role="producer">
     </AssociatedOrg>
     <ReleaseDate>2011-11-01</ReleaseDate>
     <Status>valid</Status>
     <ApproximateLength>PT45M</ApproximateLength>
     <Administrators>
      <Registrant>10.5237/superparty</Registrant>
     </Administrators>
    </BaseObjectData>
    <ExtraObjectMetadata>
     <SequenceInfo>
      <Parent>10.5240/9BCE-B814-BE24-6A85-AB05-Z</Parent>
      <md:DistributionNumber domain="eidr.org">4</md:DistributionNumber>
      <md:HouseSequence domain="eidr.org">704</md:HouseSequence></SequenceInfo>
     </SequenceInfo>
    </ExtraObjectMetadata>
   </Episode>
  </Create>
 </Operation>
</Request>
```

The above is all that is needed with a Request with Content-type of text/xml.

Sample XML of a successful Registration Response for an immediate response is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0">
 <Status><Code>0</Code><Type>success</Type></Status>
 <RequestStatus>
  <Token>1375551600399000001</Token>
 </RequestStatus>
 <RequestStatusResults>
  <CurrentSize>1</CurrentSize>
  <TotalMatches>1</TotalMatches>
  <OperationStatus>
   <Token>1375551600399000001</Token><Status><Code>0</Code>
   <Type>success</Type></Status><ID>10.5240/EAFE-C1E8-F6F5-FA04-D85B-Q</ID>
  </OperationStatus>
 </RequestStatusResults>
```

```
</Response>
```

Note that the new content ID is returned in the above response.

Sample XML of a Registration response to a non-immediate request:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0">
 <Status><Code>0</Code><Type>success</Type></Status>
 <RequestStatus><Token>1330466364470000009</Token></RequestStatus>
 <RequestStatusResults>
  <CurrentSize>1</CurrentSize><TotalMatches>1</TotalMatches>
  <BatchStatus><Code>1</Code><Type>batch received</Type></BatchStatus>
 </RequestStatusResults>
</Response>
```

Sample XML of a Registration request to Add a Lightweight Relationship (in this case a Promotion) to a record:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.eidr.org/schema"
xmlns:doi="http://www.doi.org/2010/DOISchema"
xmlns:md="http://www.movielabs.com/schema/md/v2.1/md "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Operation>
  <AddRelationship type="PromotionalRelationship">
   <ID>10.5240/4ED7-DCD7-4AF1-5545-64D5-6</ID>
   <PromotionInfo>
    <ID>10.5240/0F75-E736-22B5-562E-1867-S</ID>
    <PromotionClass>Infomercial</PromotionClass>
    </PromotionInfo>
  </AddRelationship>
 </Operation>
</Request>
```

Sample XML of a Registration request to Alias a record:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.eidr.org/schema"
xmlns:doi="http://www.doi.org/2010/DOISchema"
xmlns:md="http://www.movielabs.com/schema/md/v2.1/md "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Operation>
  <Alias>
   <ID>10.5240/A868-A057-CA54-B31E-DEDE-8</ID>
   <TargetID>10.5240/5364-C582-B6EA-25C4-AE37-E</TargetID>
  </Alias>
 </Operation>
</Request>
```

Sample XML of a Registration request to Delete a record (which is the same as aliasing it to the EIDR tombstone object):

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.eidr.org/schema"
xmlns:doi="http://www.doi.org/2010/DOISchema"
xmlns:md="http://www.movielabs.com/schema/md/v2.1/md "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Operation>
  <Delete>
```

```
    <ID>10.5240/4ED7-DCD7-4AF1-5545-64D5-6</ID>
   </Delete>
  </Operation>
</Request>
```

2.3.2 STATUS LOOKUP SERVICE

This API allows you to look up registration tokens associated with your user or Party.

The following describes the GET method:

Name	Value	
URL	One of the following: https://<*host*>.eidr.org/EIDR/status/token/<*token*> https://<*host*>.eidr.org/EIDR/status/user/<*user ID*> https://<*host*>.eidr.org/EIDR/status/registrant/<*Party ID*>	
Method	HTTP GET	
Encoding	None	
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".
HTTP GET Parameters		
Name	**Type**	**Notes**
pageNumber	Positive integer	Required. Specifies the current page number among the many pages the response could span. Also see pageSize parameter.
pageSize	Positive integer	Required. Specifies the number of records per page that the response is divided into.

Table 10: Status Lookup Service Request

The following describes the POST method, which supports filters for token status and times:

Name		Value
URL		https://<*host*>.eidr.org/EIDR/status/
Method		HTTP POST
Encoding		text/xml or multipart/form-data
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".

HTTP POST Parameters		
Name	**Type**	**Notes**
statusrequest	XML	Refer to schema for XML.

Table 11: Status Lookup Service Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type		See section "HTTP Response Headers"
Data		
status	XML	Refer to schema for XML.

Table 12: Status Lookup Service Response

An example of a status lookup GET request is as follows:

```
GET https://<host>.eidr.org/EIDR/status/token/11986584321?pageNumber=1
&pageSize=25 HTTP/1.1
Accept: text/xml
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
```

An example of the response to a status lookup request is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8


<Response XML goes here>
```

Here is a sample XML response to a token status request for a single immediate-response operation that resulted in a duplicate that is *not* considered a perfect match:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0">
 <Status><Code>0</Code><Type>success</Type></Status>
 <RequestStatus><Token>13046207826710010 23</Token></RequestStatus>
 <RequestStatusResults><CurrentSize>1</CurrentSize>
  <TotalMatches>1</TotalMatches>
  <OperationStatus>
   <Token>13046207826710010 23</Token>
   <Status><Code>1</Code><Type>duplicate</Type></Status>
   <Duplicate><ID>10.5240/FB0D-0A93-CAD6-8E8D-80C2-4</ID></Duplicate>
  </OperationStatus>
 </RequestStatusResults>
</Response>
```

If, upon further inspection, this does not appear to be a duplicate then this operation could be resubmitted in non-immediate mode to generate a manual review that would permit this registration. (Alternatively, the request could be resubmitted with more optional metadata provided that might better distinguish it.)

Here is a sample XML response to a batch token status request with two successful registration operations:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0">
 <Status><Code>0</Code><Type>success</Type></Status>
 <RequestStatus><Token>1326853306619001323</Token></RequestStatus>
 <RequestStatusResults>
 <CurrentSize>2</CurrentSize><TotalMatches>2</TotalMatches>
 <BatchStatus><Code>2</Code><Type>batch queued</Type></BatchStatus>
 <OperationStatus><Token>1326853306929001324</Token>
  <Status><Code>0</Code><Type>success</Type></Status>
 </OperationStatus>
 <OperationStatus><Token>1326853306930001325</Token>
  <Status><Code>0</Code><Type>success</Type></Status></OperationStatus>
 </RequestStatusResults>
</Response>
```

Here is a sample POST XML request for all tokens for successful registrations for a Party after a certain date:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.eidr.org/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Operation>
  <StatusRequest>
   <Registrant>10.5237/F012-89FD</Registrant>
   <Status>success</Status>
   <After>2013-02-08T00:00:00</After>
   <PageNumber>1</PageNumber><PageSize>2</PageSize>
  </StatusRequest>
 </Operation>
</Request>
```

Here is the XML response for the request:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0">
 <Status><Code>0</Code><Type>success</Type></Status>
 <RequestStatus>
  <Registrant>10.5237/superparty</Registrant>
  <Status>success</Status><After>2013-02-08T00:00:00Z</After>
  <PageNumber>1</PageNumber><PageSize>2</PageSize>
 </RequestStatus>
 <RequestStatusResults>
  <CurrentSize>2</CurrentSize>
  <TotalMatches>1955</TotalMatches>
  <OperationStatus>
   <Token>1360398065568701737</Token>
   <Status><Code>0</Code><Type>success</Type></Status>
   <ID>10.5240/FCE4-98F2-29EA-CE47-90BF-O</ID>
   </OperationStatus>
  <OperationStatus>
   <Token>1360398065574701738</Token>
   <Status><Code>0</Code><Type>success</Type></Status>
   <ID>10.5240/C4FC-B2AE-8C0B-B251-B4F3-0</ID>
  </OperationStatus>
 </RequestStatusResults>
</Response>
```

2.3.3 RESOLUTION SERVICE

This service resolves ID's the EIDR content database.

Name	Value	
URL	One of the following: https://<*host*>.eidr.org/EIDR/object/<*asset ID*> https://<*host*>.eidr.org/EIDR/object/?altId=<*alternate ID*>	
Method	HTTP GET	
Encoding	None	
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".
HTTP Parameters		
Name	**Type**	**Notes**
type	Enumeration: `Full`, `SelfDefined`, `Inherited`, `Simple`, `Provenance`, `DOIKernel`, `LinkedAlternateID`	Required. Refer to the ***EIDR Registry User's Guide*** for complete details.
followAlias	`true` or `false`	Required. Refer to the ***EIDR EIDR Registry User's Guide*** for complete details.
altId		Optional. Alternate ID. If there are multiple matches, an invalid request is returned with the number of matches. If there are no matches, a bad ID error is returned.
altIdType	Enumeration: See the schema for details.	Optional. Alternate ID type. Used to filter which AlternateIDs are returned. By default all are returned.
altIdDomain	A domain such as `studio.com`	Optional. Alternate ID domain. Used to filter which AlternateIDs are returned. By default all are returned. The parameter value "null" will indicate to return AlternateIDs where the attribute is missing.
altIdRelation	Enumeration: See the schema for details.	Optional. Alternate ID relation. The parameter value "null" will indicate to return AlternateIDs where the attribute is missing. By default relation must be missing or IsSameAs; altIdRelation=all will indicate to try to resolve assets with that altId with any relation.

Table 13: Resolution Service Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type		See section “HTTP Response Headers”
Data		
Results	XML	Refer to schema for XML.

Table 14: Resolution Service Response

An example of a resolution request is as follows:

```
GET https://<host>.eidr.org/EIDR/object/10.5240/<asset ID suffix>?type=Full
&followAlias=false HTTP/1.1
Accept: text/xml
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
```

An example of the response to a resolution request is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8

<Response XML goes here>
```

Here is a sample XML response to a resolution request for the Simple view of a root object:

```
<?xml version="1.0" encoding="UTF-8"?>
<SimpleMetadata xmlns="http://www.eidr.org/schema">
 <ID>10.5240/C840-E543-A58F-5C59-1B1C-T</ID>
 <StructuralType>Performance</StructuralType>
 <ReferentType>Movie</ReferentType>
 <ResourceName titleClass="release" lang="en">Avatar</ResourceName>
 <PrimaryLanguage type="primary">
  <Language>en</Language>
  <Manifestation>audio</Manifestation>
 </PrimaryLanguage>
 <ReleaseDate>2009</ReleaseDate>
 <Status>valid</Status>
</SimpleMetadata>
```

Here is a sample XML response to a resolution request for the Provenance view of an object:

```
<ProvenanceMetadata>
 <ID>10.5240/C44C-4039-2C9C-5D75-2174-D</ID>
 <IssueNumber>15</IssueNumber>
 <Status>valid</Status>
 <Administrators>
  <Registrant>10.5237/superparty</Registrant>
 </Administrators>
 <CreationDate>2010-12-17T07:23:23Z</CreationDate>
 <LastModificationDate>2016-10-11T18:34:24Z</LastModificationDate>
 <PublicationDate>2016-10-11T21:39:15.162Z</PublicationDate>
</ProvenanceMetadata>
```

Here is a sample XML response to a resolution request for the DOIKernel view of a root object:

```
<?xml version="1.0" encoding="UTF-8"?>
<kernelMetadata xmlns="http://www.doi.org/2010/DOISchema">
 <referentDoiName>10.5240/4DDF-A111-8543-E67B-58F6-2</referentDoiName>
 <primaryReferentType>Creation</primaryReferentType>
 <registrationAgencyDoiName>10.1000/ra-5</registrationAgencyDoiName>
 <issueDate>2013-10-03</issueDate>
 <issueNumber>8</issueNumber>
 <referentCreation>
  <name primaryLanguage="en"><value>Ben-Hur</value><type>Title</type></name>
  <identifier>
   <nonUriValue>10.5240/4DDF-A111-8543-E67B-58F6-2</nonUriValue>
   <uri
returnType="text/html">https://ui.eidr.org/view/content?id=10.5240/4DDF-A111-
8543-E67B-58F6-2</uri>
   <uri returnType="application/xml">https://doi.org/10.5240/4DDF-A111-8543-
E67B-58F6-2</uri>
   <type>EidrContentID</type>
  </identifier>
  <identifier>
   <nonUriValue>0000-0002-E823-0000-0-0000-0000-3</nonUriValue>
   <uri returnType="text/html">http://www.isan.org/lookup/0000-0002-E823-0000-
0-0000-0000-3</uri>
   <type>ISAN</type></identifier>
  <identifier>
   <nonUriValue>2009218</nonUriValue>
   <type validNamespace="warnerbros.com/MPM">ProprietaryIdentifier</type>
  </identifier>
 <structuralType>Abstraction</structuralType>
 <mode>Audio</mode><mode>Visual</mode>
 <character>Language</character>
 <character>Image</character>
 <type>Film</type>
 <principalAgent>
   <name><value>Metro-Goldwyn-Mayer</value><type>Name</type></name>
   <identifier>
    <value>10.5237/169B-EDEB</value><type>EIDRPartyID</type>
   </identifier>
   <role>CorporateCreator</role>
 </principalAgent>
 <principalAgent>
  <name><value>sam zimbalist</value><type>Name</type></name>
  <identifier>
   <value>10.5237/03E2-6787</value>
   <type>EIDRPartyID</type>
  </identifier>
  <role>CorporateCreator</role>
 </principalAgent>
 <principalAgent>
  <name><value>William Wyler</value><type>Name</type></name>
  <role>Director</role>
 </principalAgent>
 <principalAgent>
  <name><value>Charlton Heston</value>
  <type>Name</type></name>
```

```
  <role>Actor</role>
 </principalAgent>
 <principalAgent>
  <name><value>Jack Hawkins</value>
  <type>Name</type></name>
  <role>Actor</role>
 </principalAgent>
 <linkedCreation>
  <identifier>
   <nonUriValue>10.5240/6FC2-CD1E-EA8B-A2DC-BE36-O</nonUriValue>
   <uri
returnType="text/html">https://ui.eidr.org/view/content?id=10.5240/6FC2-CD1E-
EA8B-A2DC-BE36-O</uri>
   <uri returnType="application/xml">https://doi.org/10.5240/6FC2-CD1E-EA8B-
A2DC-BE36-O</uri>
   <type>EidrContentID</type>
  </identifier>
  <linkedCreationRole>Edit</linkedCreationRole>
 </linkedCreation>
 <linkedCreation>
  <identifier>
   <nonUriValue>10.5240/4BED-FDFC-C469-C7F7-1A05-2</nonUriValue>
   <uri
returnType="text/html">https://ui.eidr.org/view/content?id=10.5240/4BED-FDFC-
C469-C7F7-1A05-2</uri>
   <uri returnType="application/xml">https://doi.org/10.5240/4BED-FDFC-C469-
C7F7-1A05-2</uri>
   <type>EidrContentID</type>
  </identifier>
  <linkedCreationRole>Edit</linkedCreationRole>
 </linkedCreation>
 </referentCreation>
</kernelMetadata>
```

Here is a sample XML response to a resolution request for the SelfDefined view of a child object (Season):

```
<?xml version="1.0" encoding="UTF-8"?>
<SelfDefinedMetadata xmlns="http://www.eidr.org/schema"
xmlns:md="http://www.movielabs.com/schema/md/v2.1/md"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <BaseObjectData>
  <ID>10.5240/C44C-4039-2C9C-5D75-2174-D</ID>
  <StructuralType>Abstraction</StructuralType>
  <ReferentType>Season</ReferentType>
  <ResourceName lang="en" systemGenerated="true" titleClass="series
numeric">Seinfeld: Season 9</ResourceName>
  <AssociatedOrg idType="EIDRPartyID" organizationID="10.5237/FBF8-C3CD"
role="producer">
   <md:DisplayName>Castle Rock Entertainment</md:DisplayName>
<md:AlternateName>Castle Rock</md:AlternateName>
  </AssociatedOrg>
  <ReleaseDate>1997-09-25</ReleaseDate>
  <Status>valid</Status>
  <ApproximateLength>PT30M</ApproximateLength>
  <AlternateID xsi:type="IVA">358632</AlternateID>
  <AlternateID domain="spe.sony.com/MPM"
xsi:type="Proprietary">T5004198000</AlternateID>
```

```
  <AlternateID domain="spe.sony.com/ProductID"
xsi:type="Proprietary">20148</AlternateID>
  <AlternateID domain="nbcuni.com/sgenno"
xsi:type="Proprietary">359644</AlternateID>
  <Administrators>
   <Registrant>10.5237/superparty</Registrant>
  </Administrators>
 </BaseObjectData>
 <ExtraObjectMetadata>
  <SeasonInfo>
   <Parent>10.5240/301C-0DFA-B184-5448-BB3E-I</Parent>
   <EndDate>1998-05-14</EndDate>
   <NumberRequired>true</NumberRequired>
   <DateRequired>false</DateRequired>
   <OriginalTitleRequired>true</OriginalTitleRequired>
   <SequenceNumber>9</SequenceNumber>
  </SeasonInfo>
  </ExtraObjectMetadata>
</SelfDefinedMetadata>
```

Here is a sample XML response to a resolution request for the Inherited view of the above child object (Season):

```
<InheritedMetadata>
 <BaseObjectData><ID>10.5240/C44C-4039-2C9C-5D75-2174-D</ID>
  <Mode>AudioVisual</Mode>
  <OriginalLanguage mode="Audio" type="primary">en</OriginalLanguage>
  <CountryOfOrigin>US</CountryOfOrigin>
  <Credits>
   <Actor><md:DisplayName>Jerry Seinfeld</md:DisplayName></Actor>
   <Actor><md:DisplayName>Jason Alexander</md:DisplayName></Actor>
   <Actor><md:DisplayName>Julia Louis-Dreyfus</md:DisplayName></Actor>
   <Actor><md:DisplayName>Michael Richards</md:DisplayName></Actor>
  </Credits>
 </BaseObjectData>
</InheritedMetadata>
```

Here is a sample XML response to a resolution request for the LinkedAlternateID view:

```
<AlternateIDs xmlns="http://www.eidr.org/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ID>10.5240/C44C-4039-2C9C-5D75-2174-D</ID>
  <AlternateID xsi:type="IVA">358632</AlternateID>
  <AlternateID domain="spe.sony.com/MPM"
xsi:type="Proprietary">T5004198000</AlternateID>
  <AlternateID domain="spe.sony.com/ProductID"
xsi:type="Proprietary">20148</AlternateID>
  <AlternateID domain="nbcuni.com/sgenno"
xsi:type="Proprietary">359644</AlternateID>
</AlternateIDs>
```

2.3.4 MATCH SERVICE

This service is used to find content matches prior to registration using the proposed registration data as a kind of query. Note that fields that are not used to distinguish records during de-duplication are ignored in the query.

Name	Value	
URL	https://<*host*>.eidr.org/EIDR/match/	
Method	HTTP POST	
Encoding	text/xml or multipart/form-data	
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section “HTTP Request Headers”.
Authorization		The Party used must have a Registrant role. See section “HTTP Request Headers” for details.
Immediate-Response		Must be "true". See section “HTTP Request Headers” for details.
HTTP POST Parameters		
Name	**Type**	**Notes**
batch	XML	Refer to schema for XML.

Table 15: Match Service Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type	Internet Media Type (MIME)	See section “HTTP Response Headers”
Data		
Response	XML	Refer to schema for XML.

Table 16: Match Service Response

An example of a match request is as follows:

```
POST https://<host>.eidr.org/EIDR/match/ HTTP/1.1
Accept: text/xml
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
Immediate-Response: true
Content-Type: multipart/form-data; boundary=---314159
---314159
Content-Disposition: form-data; name=batch
```

```
Content-Transfer-Encoding: binary
<Request XML goes here>
---314159--
```

The format of the response to a match request is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8

<Response XML goes here>
```

The request is identical to a Registration. For details see “Registration service”.

The response is a modified version of the Registration Response that is more like a query. The differences are as follows:

- There is no “duplicate” Operation Status Code. An Operation Status of “success” indicates that there are match results.
- There is no repetition of the ID in the case of perfect duplicates. Therefore de-duplication scores are always returned. A perfect duplicate would be a single duplicate with a score that is greater than or equal to the high threshold for the object type. One or more low threshold duplicates would go to manual de-duplication. More than one high duplicates also go through manual de-duplication. No duplicates indicates the registration would succeed (assuming no changes in the database related to this record).
- The data validation rule that disallows duplicate episode numbers is not enforced so that matches may be identified.

An example of the response XML to a match request for a non-Series Base object (such as a Movie or OTO) that is identical to an existing record is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0">
 <Status><Code>0</Code><Type>success</Type></Status>
 <RequestStatus><Token>13412618688560004 56</Token></RequestStatus>
<RequestStatusResults>
<CurrentSize>1</CurrentSize><TotalMatches>1</TotalMatches>
  <OperationStatus><Token>13412618688560004 56</Token>
   <Status><Code>0</Code><Type>success</Type></Status>
   <Duplicate score="100" lowThreshold="55" highThreshold="85">
    <ID>10.5240/826B-820C-28CF-2019-43FD-Q</ID>
   </Duplicate>
  </OperationStatus>
 </RequestStatusResults>
</Response>
```

An example of the response XML to a match request for a record that has no duplicates objects according to the de-duplication system:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0.1">
<Status><Code>0</Code><Type>success</Type></Status>
<RequestStatus>
 <Token>13872482262680221 64</Token>
```

```
</RequestStatus>
<RequestStatusResults>
 <CurrentSize>1</CurrentSize><TotalMatches>1</TotalMatches>
  <OperationStatus><Token>1387248226268022164</Token>
   <Status><Code>0</Code>
    <Type>success</Type>
   </Status>
  </OperationStatus>
 </RequestStatusResults>
</Response>
```

This indicates that this is a gap record and would create a new record if registered.

2.3.5 QUERY SERVICE

This service queries the EIDR content database.

Name	Value	
URL	https://<*host*>.eidr.org/EIDR/query/	
Method	HTTP POST	
Encoding	text/xml or multipart/form-data	
HTTP Parameters		
Name	**Type**	**Notes**
type	Enumeration: ID simple (or SimpleMetadata)	Optional. ID returns only the ID of the content record. If not provided then the default is SimpleMetadata. See examples below.
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Accept-Encoding		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".
HTTP POST Parameters		
Name	**Type**	**Notes**
query	XML	Refer to schema for XML.

Table 17: Query Service Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type	Internet Media Type (MIME)	See section "HTTP Response Headers"
Data		
Queryresults	XML	Refer to schema for XML.

Table 18: Query Service Response

An example of a query request using multipart/form-data is as follows:

```
POST https://<host>.eidr.org/EIDR/query/ HTTP/1.1
Accept: text/xml
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
Content-Type: multipart/form-data; boundary=---314159
---314159
```

```
Content-Disposition: form-data; name=query
Content-Transfer-Encoding: binary
<Request XML goes here>
---314159--
```

A format of the response to a query request is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8

<Response XML goes here>
```

Query expressions are used when making a query with the POST method. Refer to the ***EIDR Registry User's Guide*** for the query expression grammar.

Here is a sample XML for a content record query:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.eidr.org/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Operation>
  <Query>
   <Expression>(/FullMetadata/BaseObjectData/ResourceName IS "Fight
Club")</Expression>
   <PageNumber>1</PageNumber>
   <PageSize>3</PageSize>
  </Query>
 </Operation>
</Request>
```

An example of the response XML to a default (SimpleMetadata) query request is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema"
xmlns:md="http://www.movielabs.com/schema/md/v2.1/md"
xmlns:doi="http://www.doi.org/2010/DOISchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
 <Status>
  <Code>0</Code>
  <Type>success</Type>
 </Status>
 <Query>
  <Expression>(/FullMetadata/BaseObjectData/ResourceName "II") AND
/FullMetadata/BaseObjectData/ReferentType "movie"</Expression>
  <PageNumber>1</PageNumber>
  <PageSize>20</PageSize>
 </Query>
 <QueryResults>
  <CurrentSize>20</CurrentSize>
  <TotalMatches>230</TotalMatches>
  <SimpleMetadata>
   <ID>10.5240/0B20-3C24-2838-91EB-08CC-N</ID>
   <StructuralType>Performance</StructuralType>
   <ReferentType>Movie</ReferentType>
```

```
   <ResourceName titleClass="release" lang="en">Young and Dangerous
II</ResourceName>
   <PrimaryLanguage type="primary">
    <Language>en</Language>
    <Manifestation>audio</Manifestation>
   </PrimaryLanguage>
   <ReleaseDate>1996</ReleaseDate>
   <Status>valid</Status>
  </SimpleMetadata
…  [2 other results]
 </QueryResults>
</Response>
```

An example of the response XML to a query request with type=id is as follows:

```
<?xml version="1.0" encoding="UTF-8"?><Response
xmlns="http://www.eidr.org/schema"
version="2.0.5"><Status><Code>0</Code><Type>success</Type></Status>
 <Query><Expression>/FullMetadata/BaseObjectData/ResourceName IS "Fight
Club"</Expression>
 <PageNumber>1</PageNumber><PageSize>10</PageSize></Query>
 <QueryResults><CurrentSize>8</CurrentSize><TotalMatches>8</TotalMatches>
  <ID>10.5240/F19F-D2DB-43F5-9B97-ED62-S</ID>
  <ID>10.5240/6865-ACEF-7D09-DAD3-7B5A-F</ID>
  <ID>10.5240/5CC7-E9D7-C6FC-991A-AE44-0</ID>
  <ID>10.5240/BE8F-D145-42D7-1E9B-E670-L</ID>
  <ID>10.5240/0517-8D84-F801-2128-2995-K</ID>
  <ID>10.5240/1FA1-D212-2A2A-0247-4735-H</ID>
  <ID>10.5240/E570-FE04-F9E5-3BB4-89A9-Y</ID>
  <ID>10.5240/333B-2034-D88E-E735-69E0-A</ID>
 </QueryResults>
</Response>
```

2.3.6 GRAPH TRAVERSAL SERVICE

This service gets information for the inheritance hierarchy of an ID the EIDR content database.

Name	Value	
URL	https://<*host*>.eidr.org/EIDR/object/graph/	
Method	HTTP POST	
Encoding	text/xml or multipart/form-data	
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section “HTTP Request Headers”.
Authorization		See section “HTTP Request Headers”.
HTTP Parameters		
Name	**Type**	**Notes**
extendedFamily	Boolean	Optional. When true, traversals will include "dependents" of descendants as well as the ordinary descendants. Default is “false”.
HTTP POST Parameters		
Name	**Type**	**Notes**
graphrequest	XML	Refer to api.xsd for schema.

Table 19: Graph Traversal Service Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type		See section “HTTP Response Headers”
Data		
Graph	XML	Refer to schema for XML.

Table 20: Graph Traversal Service Response

The format of a graph traversal request using multipart/form-data is as follows:

```
POST https://<host>.eidr.org/EIDR/object/graph/ HTTP/1.1
Accept: text/xml
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
Immediate-Response: true
Content-Type: multipart/form-data; boundary=---314159
---314159
Content-Disposition: form-data; name=graphrequest
Content-Transfer-Encoding: binary
<Request XML goes here>
---314159--
```

The format of the response to a traversal request is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8


<Response XML goes here>
```

An example of the request XML for a graph traversal is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="http://www.eidr.org/schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Operation>
  <GetRemotestAncestor>
   <ID>10.5240/8B55-F9AA-007F-B18E-C000-6</ID>
  </GetRemotestAncestor>
 </Operation>
</Request>
```

An example of the response XML to a graph traversal request is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema" version="2.0.9">
 <Status><Code>0</Code><Type>success</Type></Status>
 <SimpleMetadata>
  <ID>10.5240/FB0D-0A93-CAD6-8E8D-80C2-4</ID>
  <StructuralType>Abstraction</StructuralType>
  <ReferentType>Movie</ReferentType>
  <ResourceName titleClass="release" lang="en">Gone with the
Wind</ResourceName>
  <OriginalLanguage mode="Audio" type="primary">en</OriginalLanguage>
  <ReleaseDate>1939-12-15</ReleaseDate>
  <Status>valid</Status>
 </SimpleMetadata>
 <GenerationsAbove>1</GenerationsAbove>
</Response>
```

2.3.7 MODIFICATION BASE SERVICE

This service reads the Modification Base for ID's the EIDR content database.

Name	Value	
URL	https://<*host*>.eidr.org/EIDR/object/modificationbase/	
Method	HTTP GET	
Encoding	None	
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".
HTTP GET Parameters		
Name	**Type**	**Notes**
type	Controlled vocabulary: `CreateBasic`, `CreateSeries`, `CreateSeason`, `CreateEpisode`, `CreateClip`, `CreateCompilation`, `CreateComposite`, `CreateEdit`, `CreateManifestation`, `CreateInteractive`	Required, of type eidr:creationType. See the common.xsd schema for details.

Table 21: Modification Base Service Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type		See section "HTTP Response Headers"
Data		
Results	XML	Refer to schema for XML.

Table 22: Modification Base Service Response

An example of a Modification Base request is as follows:

```
GET https://<host>.eidr.org/EIDR/object/modificationbase/10.5240/<asset ID suffix>?type=CreateBasic
HTTP/1.1
Accept: text/xml
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
```

An example of the response to a Modification Base request is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8
<Response XML goes here>
```

An example of the response XML to a Modification Base request is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema"
xmlns:md="http://www.movielabs.com/schema/md/v2.1/md"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
 <Status><Code>0</Code><Type>success</Type></Status>
 <Basic>
  <BaseObjectData>
   <StructuralType>Performance</StructuralType><Mode>AudioVisual</Mode>
   <ReferentType>Movie</ReferentType>
   <ResourceName titleClass="release" lang="en">Ben-Hur</ResourceName>
   <OriginalLanguage mode="Audio" type="primary">en</OriginalLanguage>
   <AssociatedOrg idType="EIDRPartyID" organizationID="10.5237/169B-EDEB"
role="producer">
    <md:DisplayName>Metro-Goldwyn-Mayer</md:DisplayName>
    <md:AlternateName>MGM</md:AlternateName>
   </AssociatedOrg>
   <AssociatedOrg idType="EIDRPartyID" organizationID="10.5237/03E2-6787"
role="producer">
    <md:DisplayName>sam zimbalist</md:DisplayName>
   </AssociatedOrg>
   <ReleaseDate>1959</ReleaseDate>
   <CountryOfOrigin>US</CountryOfOrigin>
   <Status>valid</Status>
   <ApproximateLength>PT3H32M</ApproximateLength>
   <AlternateID xsi:type="ISAN">0000-0002-E823-0000-0-0000-0000-3</AlternateID>
   <Administrators>
    <Registrant>10.5237/superparty</Registrant>
   </Administrators>
   <Credits>
    <Director><md:DisplayName>William Wyler</md:DisplayName></Director>
    <Actor><md:DisplayName>Charlton Heston</md:DisplayName></Actor>
    <Actor><md:DisplayName>Jack Hawkins </md:DisplayName></Actor>
   </Credits>
  </BaseObjectData>
 </Basic>
</Response>
```

For details on using the feature see the *EIDR Registry User's Guide*.

2.3.8 VIRTUAL FIELDS RETRIEVAL SERVICE

This service reads certain BaseObjectData free-text fields for ID's the EIDR content database.

Name	Value	
URL	https://<*host*>.eidr.org/EIDR/virtual/object/<*asset ID*>	
Method	HTTP GET	
Encoding	None	
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".

Table 23: Virtual Fields Retrieval Service Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type		See section "HTTP Response Headers"
Data		
Virtual Fields	XML	Refer to schema for XML.

Table 24: Virtual Fields Retrieval Service Response

An example of a virtual fields retrieval request is as follows:

```
GET https://<host>.eidr.org/EIDR/virtual/object/10.5240/<asset ID suffix>
HTTP/1.1
Accept: text/xml
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
```

An example of the response to a virtual fields retrieval request is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8

<Response XML goes here>
```

Here is a sample XML response to a virtual fields retrieval request for an object:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="http://www.eidr.org/schema">
<Status><Code>0</Code><Type>success</Type></Status>
<ID>10.5240/C840-E543-A58F-5C59-1B1C-T</ID>
<VirtualFields>
 <Full>James Cameron Sam Worthington Zoe Saldana  Avatar</Full>
 <SelfDefined>Sam Worthington Zoe Saldana Avatar  James Cameron</SelfDefined>
</VirtualFields></Response>
```

2.3.9 PARTY RESOLUTION SERVICE

This service performs resolutions of ID's in the EIDR Party database.

Name		Value
URL		https://<*host*>.eidr.org/EIDR/party/resolve/?type=[doi | full]
Method		HTTP GET
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".

Table 25: Resolve Party Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type		See section "HTTP Response Headers"
Data		
Response	XML	An instance of the eidr:AdminResponse XML schema in case of error, or the DOI Kernel Metadata for parties if the type specified is "`doiKernel`", or an instance of eidr:Party XML the type specified is "`full`"
HTTP GET Parameters		
Name	**Type**	**Notes**
type	Enumeration: `full`	Required.

Table 26: Resolve Party Response

An example of a Party resolution request is as follows:

```
GET https://<host>.eidr.org/EIDR/party/resolve/10.5237/<Party ID
suffix>?type=full HTTP/1.1
Accept: text/xml
Authorization: Eidr
10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
```

An example of resolve Party response is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8

<eidr:Party XML>
```

An example of the XML of a Party `full` resolution request is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Party xmlns="http://www.eidr.org/schema"
xmlns:md="http://www.movielabs.com/schema/md/v2.1/md">
 <ID>10.5237/53C2-B532</ID>
 <PartyName organizationID="Principal Agent">
  <md:DisplayName>Republic Pictures</md:DisplayName>
 </PartyName>
 <ContactInfo>
  <md:Name>Republic Pictures</md:Name>
  <md:PrimaryEmail>not_available@eidr.org</md:PrimaryEmail>
 </ContactInfo>
 <Active>true</Active>
 <PartyAccountName>53C2-B532</PartyAccountName>
 <AllowedRoles>PrincipalAgent</AllowedRoles>
</Party>
```

An example of the XML of a Party `doi` resolution request is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<kernelMetadata xmlns='http://www.doi.org/2010/DOISchema'>
 <referentDoiName>10.5237/AD45-F060</referentDoiName>
 <primaryReferentType>Party</primaryReferentType>
 <registrationAgencyDoiName>10.1000/ra-5</registrationAgencyDoiName>
 <issueDate>0001-01-01</issueDate>
 <issueNumber>0</issueNumber>
 <referentParty>
 <name>
  <value>Walt Disney Studios Home Entertainment</value>
  <type>PrincipalName</type>
 </name>
 <structuralType>Organization</structuralType>
 <associatedPartyRole>CorporateCreator</associatedPartyRole>
 </referentParty>
</kernelMetadata>
```

2.3.10 PARTY QUERY SERVICE

This service queries the EIDR Party database.

Name	**Value**	
URL	https://<*host*>.eidr.org/EIDR/party/query?type={ID | full}	
Method	HTTP POST	
Encoding	text/xml or multipart/form-data	
HTTP Headers		
Name	**Type**	**Notes**
Accept		See section "HTTP Request Headers".
Authorization		See section "HTTP Request Headers".
HTTP POST Parameters		
Name	**Type**	**Notes**
partyQuery	XML	An instance of one of the following XML elements: eidr:FindParties eidr:FindPartiesByName eidr:FindPartiesFromCatalog
type		ID = Party ID's only full = full party resolution data

Table 27: Query Parties Request

HTTP Headers		
Name	**Type**	**Notes**
Content-type		See section "HTTP Response Headers"
Data		
response	XML	An instance of the eidr:PartyQueryResults or eidr:AdminReponse in case of failure

Table 28: Query Parties Response

An example of a Party query request is as follows:

```
POST https://<host>.eidr.org/EIDR/party/query/ HTTP/1.1
Accept: text/xml
Authorization: Eidr 10.5238/john.doe:10.5237/A929-C667:PtEdr8lBQ45IbT1bZIkrOQ==
Content-Type: multipart/form-data; boundary=---314159
---314159
Content-Disposition: form-data; name=partyQuery
Content-Transfer-Encoding: binary
<eidr:FindPartiesFromCatalog XML goes here>
```

```
---314159--
```

An example of a Party query response is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8


<eidr:PartyQueryResults instance goes here>
```

Here is a sample XML for a Party query:

```
<?xml version="1.0" encoding="UTF-8"?>
<FindPartiesFromCatalog xmlns='http://www.eidr.org/schema'>
 <PartyName>viacom</PartyName>
 <ActiveFilter/>
 <PageNumber>0</PageNumber>
 <PageSize>0</PageSize>
</FindPartiesFromCatalog>
```

An example of the XML of a Party query response is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<PartyQueryResults xmlns="http://www.eidr.org/schema">
 <CurrentSize>4</CurrentSize>
 <TotalMatches>4</TotalMatches>
 <PartyID>10.5237/31DE-E215</PartyID>
 <PartyID>10.5237/8364-9A32</PartyID>
 <PartyID>10.5237/F1C4-FCB1</PartyID>
 <PartyID>10.5237/A59E-CC68</PartyID>
</PartyQueryResults>
```