Telestream

DIVA

REST API

Programmer's Guide

Release: 9.0

Revision: 1.0

January 9, 2024

2401091713

Copyrights and Trademark Notices

Contents

Telestream Contact Information

To obtain product information, technical support, or provide comments on this guide, contact us using our web site, email, or phone number as listed below.

Resource	Contact Information
DIVA Technical Support	Web Site: https://www.telestream.net/telestream-support/ Depending on the problem severity, we will respond to your request within 24 business hours. For P1, we will respond within 1 hour. Please see the Maintenance & Support Guide for these definitions. • Support hours for customers are Monday-Friday, 7am-6pm local time. • P1 issues for customers are 24/7.
Telestream, LLC	Web Site: www.telestream.net Sales and Marketing Email: info@telestream.net Telestream, LLC 848 Gold Flat Road, Suite 1 Nevada City, CA USA 95959
International Distributor Support	Web Site: www.telestream.net See the Telestream Web site for your regional authorized Telestream distributor.
Telestream Technical Writers	Email: techwriter@telestream.net Share comments about this or other Telestream documents.

Preface

This book gives an operational understanding of system functionality and instructions for using the DIVA REST API.

Topics

Audience

This document is intended for Installation, Administration and Operations personnel to follow all of the necessary steps to achieve full functionality of the DIVA REST API component.

Documentation Accessibility

For information about Telestream's commitment to accessibility, visit the Telestream Support Portal located at https://www.telestream.net/telestream-support/.

Related Documents

For more information, see the Telestream Core documentation set for this release located at: https://www.telestream.net/telestream-support/.

Document Updates

The following table identifies updates made to this document.

Date	**Update**
April 2022	Updated Copyright information. Updated book for release 8.2. Updated terminology to new standards.
July 2022	Migrated book to Telestream format and styles.
September 2022	Updated terminology and title page graphic. (see the *Overview* for updated terms)
October 2022	Updated book for release 8.3
November 2022	Added preface page to book. Reverted the term "Virtual Object" to "Object".
December 2022	Updated book for 8.3.1 release.
February 2023	Updated book for 9.0 release. Updated copyright dates.
March 2023	Updated book from DIVA Core to Content Manager.
June 2023	Updated book with review comments received.

Overview

Telestream recommends using the REST API rather than the previous existing APIs (that is, DIVA Enterprise Connect, DIVAS, Java and C++). Although all previous APIs will remain available, the REST API offers new and enhanced features and is integrated into DIVA and is required by the web app to function.

REST API JSON files can be downloaded from Share Point here:

https://tinyurl.com/JSON-Files

or from the Telestream DIVA Support Portal:
https://www.telestream.net/telestream-support/.

Note: To enable API backward compatibility, the term *job* has been removed and replaced with *request* used in legacy API structures and commands.

Topics

DIVA Concepts

The following information are standard DIVA concepts. Refer to the DIVA Architecture, Concepts and Glossary book for additional details.

Archive Request

DIVA stores objects; an object is a set of files referring to an asset or a clip. An object can be made of 1 file, typically MXF file or with several files like reference mov format (one video file, several audio files), or DPX format.

An object is identified by a Name and a Collection (category). YChoose whatever names for Object Name and Collection desired. DIVA only checks that the Object Name + Collection combination is unique.

In DIVA, a Collection is like a name extension and should not be confused with a Tape Group. Any name can be used for the Collection. Telestream recommends using the application or company name so we can identify who has sent a request. Should the same Object Name be used for different clips (typically hi-res and low-res), use a different Collection to distinguish those clips.

The Files parameter provides the names of the files of the object to be archived; each name can contain a relative path to the file location.

Media Name is the DIVA device used for storing the object; it can be a disk, a tape or cloud storage. Each of these devices can have multiple names based on partitioning (for example, DIVAGRID, NAS-STORAGE, TAPE_SPORTS_MAIN, TAPE_SPORTS_BKP, CLOUD_PROGRAM, CLOUD_PROMOS, and so on). The list of all Arrays and Tape Groups can be retrieved from DIVA, but it does not necessarily mean they need to be exposed to the end user. The Media can also be a Storage Plan (see the Storage Policy Manager book for details). Check with the customer and the DIVA Project Manager about which Media to expose to the end user.

The Unmanaged Storage Repository Name is the content server name where DIVA will archive from. It must be the same name as in the DIVA configuration. Confirm this with the customer or DIVA Project Manager for this list.

The Source Path Root is the File Path Root where the content objects are located. By default, DIVA will use the default File Path Root configured for that source in the DIVA configuration.

Note: The Source list can be obtained using the *GET /servers* API call.

The Quality of Service parameter can remain at the default setting.

The Priority (between 1 and 100 highest) can either remain at the default, or a value can be specified.

If the Delete From Source option check box is selected, then that parameter will delete the asset just archived from the Source Server, but only if the archive was successful.

Restore Request

The following items must be specified for a Restore Request:

- Object Name
- Object Collection
- Unmanaged Storage Repository Server Name
- The File Path Root; if empty, DIVA will take the File Path Root used during the Archive request and will overwrite the object if it already exists, unless the Do Not Overwrite option is specified.

Partial Restore

The Partial Restore parameters are the same as the Restore parameters with the following additional options:

- Offset or Timecodes (In/Out) or File List
- Partial Restore will create a new clip name because it generates a new clip created with a portion of the original clip.

Delete Request

A Delete Object Request will delete all copies of that object whether they are on disk, tape (in the tape library or external), or in the cloud. The Object Name and Object Collection must be specified.

Note: Deleting an object implies it cannot be recovered from the storage media after deletion.

Main DIVA API Calls

The following are the main DIVA API calls available and are the minimum required to implement the basic *DIVA API Workflows*:

- *POST /users/logins*
- *POST /users/logout*
- *GET /groups*
- *GET /arrays*
- *GET /object/info*
- *GET /objects/list*
- *GET /requests*
- *POST /requests/archive*
- *POST /requests/cancel*
- *POST /requests/delete*
- *POST /requests/partialRestore*
- *POST /requests/restore*
- *GET /requests/{requestId}*
- *GET /versions*

The Swagger definitions for these endpoints are located here:

https://127.0.0.1:8765/webjars/swagger-ui/index.html?urls.primaryName=data

https://127.0.0.1:8765/webjars/swagger-ui/index.html?urls.primaryName=manager

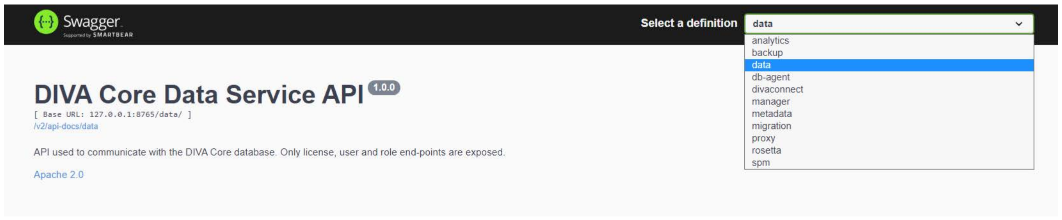
Getting Started

This chapter guides the user through getting started using the DIVA REST API.

Topics

Structure

The REST API Swagger page is home to all DIVA REST APIs. You can toggle to other APIs using the Definition at the top right side of the page.



Swagger
Supported by SMARTBEAR

Select a definition: data

- analytics
- backup
- data
- db-agent
- divaconnect
- manager
- metadata
- migration
- proxy
- rosetta
- spm

DIVA Core Data Service API 1.0.0

```
[ Base URL: 127.0.0.1:8765/data/ ]
```

/v2/api-docs/data

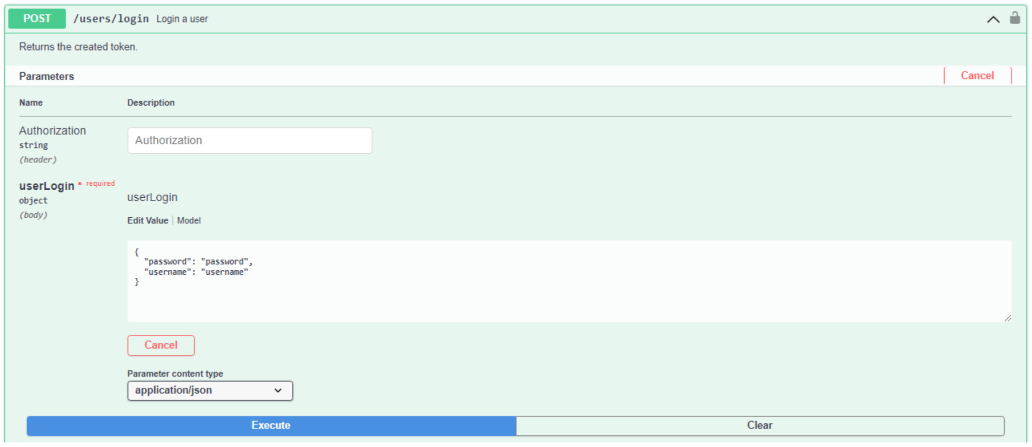
API used to communicate with the DIVA Core database. Only license, user and role end-points are exposed.

Apache 2.0

For the purposes of this document we will focus of the data and manager service required to login and submit requests, respectively.

Initial Configuration

During installation a user is created by either the DIVA Installer, or manually by an administrator. This information must be obtained from the person who created the user; all automations and API calls use these credentials. Go to the `POST users/login` endpoint and specify the login and password to log in; this is sufficient to obtain a token and proceed with other API calls.



Returns the created token.

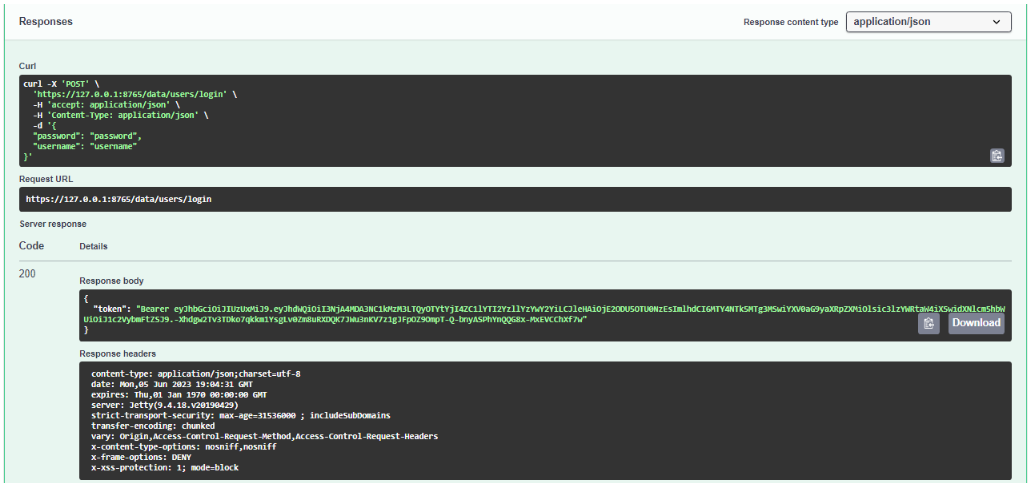
Parameters

Name	Description
Authorization string *(header)*	Authorization
userLogin * required object *(body)*	userLogin

```
{
  "password": "password",
  "username": "username"
}
```

Parameter content type: application/json

Click the Try it out button and you will receive a token. Copy the contents of the Bearer token (everything in quotes after "token") as shown in the following figure:



Responses

Response content type: application/json

Curl

```
curl -X 'POST' \
  'https://127.0.0.1:8765/data/users/login' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "password": "password",
  "username": "username"
}'
```

Request URL

```
https://127.0.0.1:8765/data/users/login
```

Server response

Code: 200

Response body

```
{
  "token": "Bearer eyJhbGciOiJIUzUxMiJ9.eyJhdWQiOiI3NjA4MDA3NC1kMzM3LTQyOTYtYjI4ZC1lYTI2YzllYzYwY2YiLCJleHAiOjE2ODU5OTU0NzEsImlhdCI6MTY4NTk5MTg3MSwiYXV0aG9yaXRpZXMiOlsic3lzYWRtaW4iXSwidXNlcm5hbW
UiOiJ1c2VybmFtZSJ9.-Xhdgw2Tv3TDko7qkkm1YsgLv0Zm8uRXDQK7JWu3nKV7z1gJFpOZ9OmpT-Q-bnyASPhYnQQG8x-MxEVCChXf7w"
}
```

Response headers

```
 content-type: application/json;charset=utf-8
 date: Mon,05 Jun 2023 19:04:31 GMT
 expires: Thu,01 Jan 1970 00:00:00 GMT
 server: Jetty(9.4.18.v20190429)
 strict-transport-security: max-age=31536000 ; includeSubDomains
 transfer-encoding: chunked
 vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
 x-content-type-options: nosniff,nosniff
 x-frame-options: DENY
 x-xss-protection: 1; mode=block
```

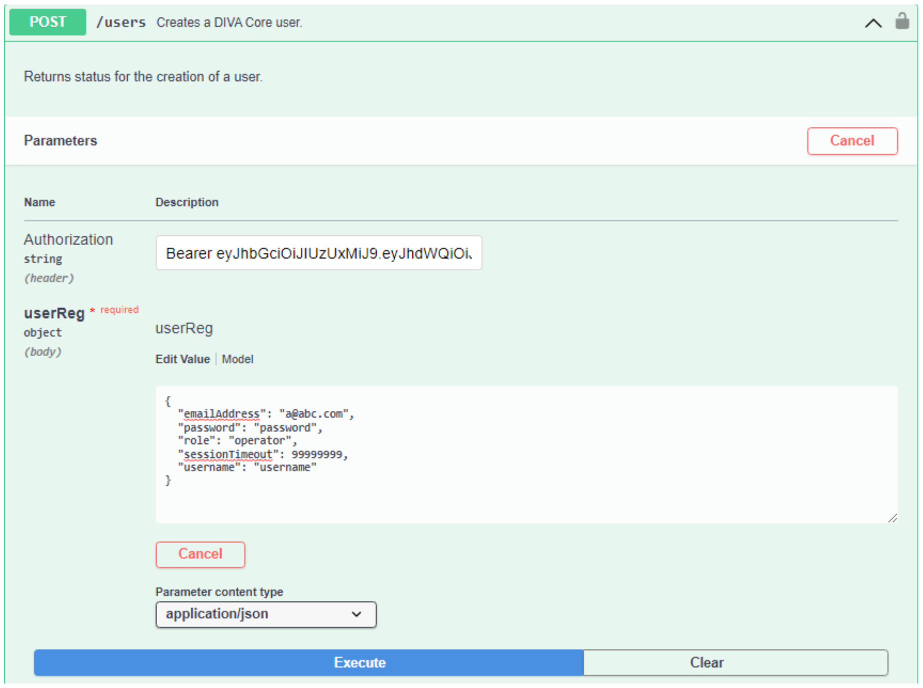
A *POST /users* request must be submitted by entering the token in the Authorization field to create a user. The user name, password, and role of the user to create must be specified (see the following figure). Optionally, you may specify an email address and a session timeout in minutes when creating a user. The session timeout is how long the user will remain logged into DIVA. You can configure it to a value between 0 and 10080 minutes. Default: 1440 minutes.

Note: Call *GET /roles* to obtain a list of possible roles.

- All DIVA GET requests require at least the user role.
- Archive, Restore (including N-Restore and Partial Restore) and Copy requests require at least the operator role.

- Change Priority, Transfer, Eject, Insert, Export and Import requests require at least the advoperator role.
- All other requests require the administrator role.

Here is an example:



Returns status for the creation of a user.

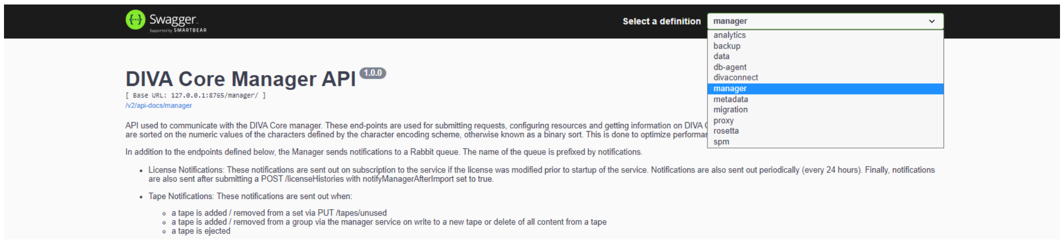
Authorization `string` *(header)*: Bearer eyJhbGciOiJIUzUxMiJ9.eyJhdWQiOi

userReg * required `object` *(body)*: userReg

```
{
  "emailAddress": "a@abc.com",
  "password": "password",
  "role": "operator",
  "sessionTimeout": 99999999,
  "username": "username"
}
```

Parameter content type: application/json

The API is now ready to be used to retrieve information from DIVA. Switch to the Manager endpoints to start using the API.



DIVA Core Manager API 1.0.0

[Base URL: 127.0.0.1:8765/manager/]
/v2/api-docs/manager

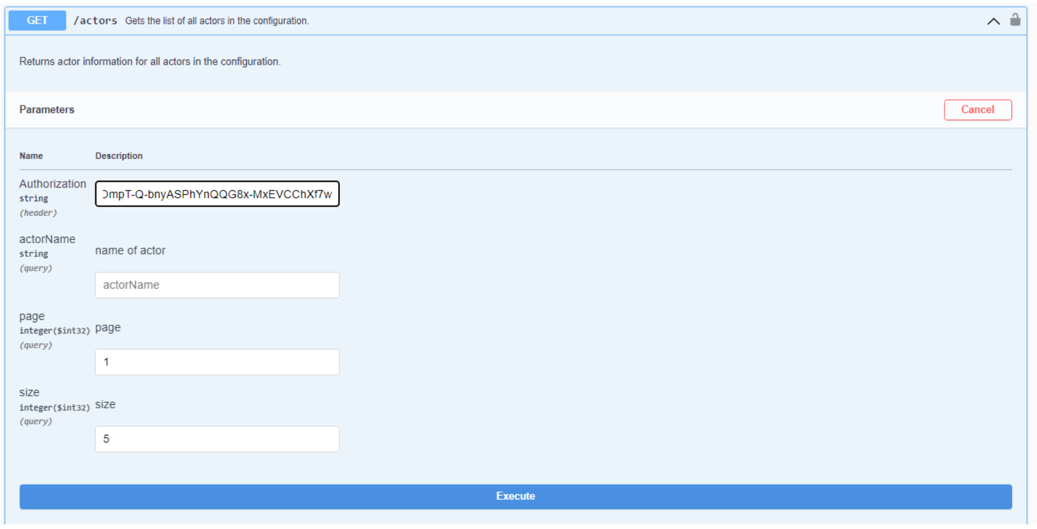
API used to communicate with the DIVA Core manager. These end-points are used for submitting requests, configuring resources and getting information on DIVA C
are sorted on the numeric values of the characters defined by the character encoding scheme, otherwise known as a binary sort. This is done to optimize performar

In addition to the endpoints defined below, the Manager sends notifications to a Rabbit queue. The name of the queue is prefixed by notifications.

- License Notifications: These notifications are sent out on subscription to the service if the license was modified prior to startup of the service. Notifications are also sent out periodically (every 24 hours). Finally, notifications are also sent after submitting a POST /licenseHistories with notifyManagerAfterImport set to true.
- Tape Notifications: These notifications are sent out when:
    - a tape is added / removed from a set via PUT /tapes/unused
    - a tape is added / removed from a group via the manager service on write to a new tape or delete of all content from a tape
    - a tape is ejected

Retrieving All Configured Actors

This figure is an example call to retrieve all configured Actors:



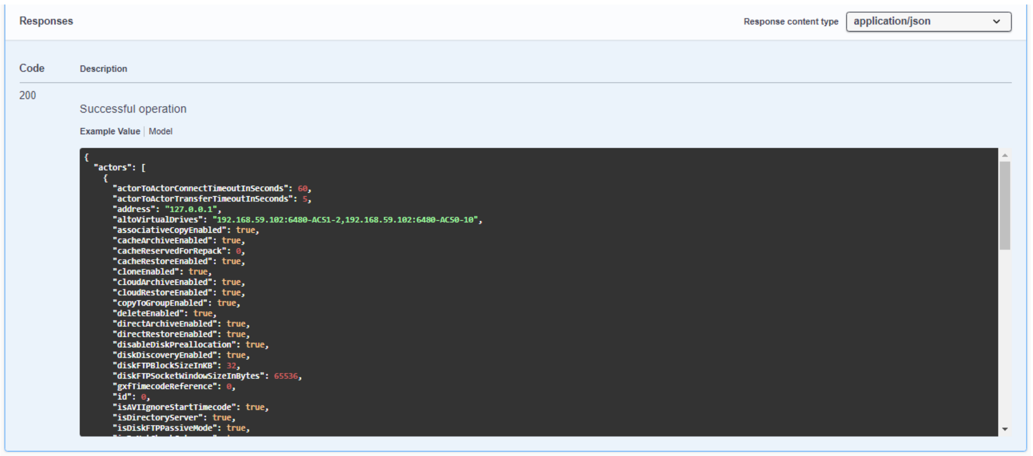
GET /actors Gets the list of all actors in the configuration.

Returns actor information for all actors in the configuration.

Parameters

Name	Description
Authorization string (header)	ͻmpT-Q-bnyASPhYnQQG8x-MxEVCChXf7w
actorName string (query)	name of actor actorName
page integer($int32) (query)	page 1
size integer($int32) (query)	size 5

This figure shows the start of the response:



Responses

Response content type: application/json

Code	Description
200	Successful operation

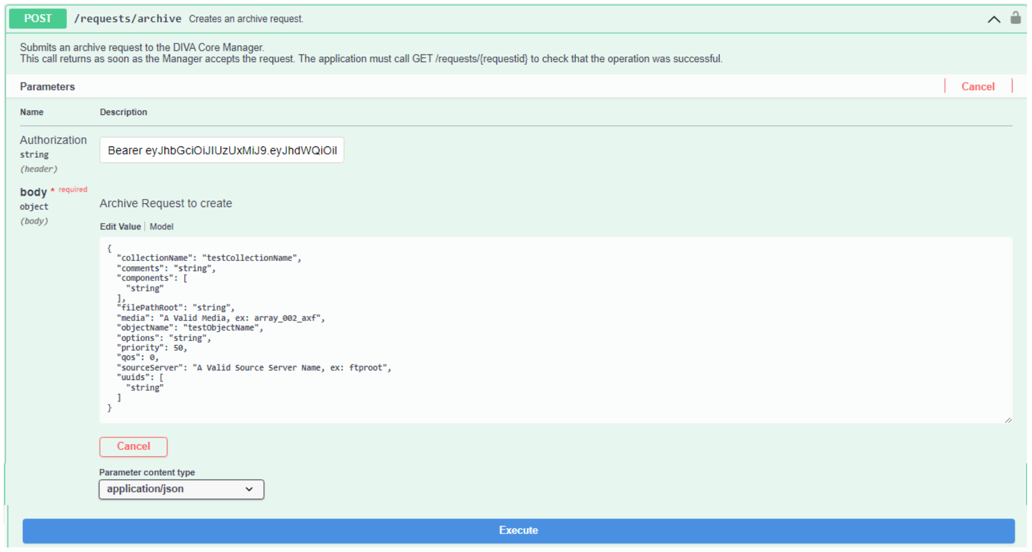
```
{
  "actors": [
    {
      "actorToActorConnectTimeoutInSeconds": 60,
      "actorToActorTransferTimeoutInSeconds": 5,
      "address": "127.0.0.1",
      "altoVirtualDrives": "192.168.59.102:6480-ACS1-2,192.168.59.102:6480-ACS0-10",
      "associativeCopyEnabled": true,
      "cacheArchiveEnabled": true,
      "cacheReservedForRepack": 0,
      "cacheRestoreEnabled": true,
      "cloneEnabled": true,
      "cloudArchiveEnabled": true,
      "cloudRestoreEnabled": true,
      "copyToGroupEnabled": true,
      "deleteEnabled": true,
      "directArchiveEnabled": true,
      "directRestoreEnabled": true,
      "disableDiskPreallocation": true,
      "diskDiscoveryEnabled": true,
      "diskFTPBlockSizeInKB": 32,
      "diskFTPSocketWindowSizeInBytes": 65536,
      "gxfTimecodeReference": 0,
      "id": 0,
      "isAVIIgnoreStartTimecode": true,
      "isDirectoryServer": true,
      "isDiskFTPPassiveMode": true,
```

Submitting a Request

To submit a request (for example, an Archive request) submit a *POST /requests/archive*. The header must contain an Authorization Key with the bearer token as the value, as depicted in this request:

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' --header 'Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJhdWQiOiI1MjM5YTcxOS1iYjAwLTQ5MWQtOGYxZi01Z
jcxM2YxZWZiMjMiLCJleHAiOjE2MjEzNTY0MDcsImlhdCI6MTYyMTI3MDAwNywiYXV
0aG9yaXRpZXMiOlsic3lzYWRtaW4iXSwidXNlcm5hbWUiOiJzeXNhZG1pbiJ9.zZiK
vEe-3JjuOsJ-CDpW_32JKRefy54-wGwra_LABmUeuIhpWGEpHnT-
Se5PXTFxvjDf2g9mgezKQIvIJzObzQ' -d '{ \
   "collectionName": "a", \
   "comments": "this is object a2", \
   "components": [ \
     "1.txt" \
   ], \
   "filePathRoot": "", \
   "media": "default", \
   "objectName": "a2", \
   "options": "", \
   "priority": 50, \
   "qos": 2, \
   "sourceServer": "wfm_ftp_sd_for_diva_test" \
 }' 'http://172.16.10.18:8765/manager/requests/archive'
```

Go to the Swagger page for the request and click on the Example Value to view the fields that must be specified for any request.



POST /requests/archive Creates an archive request.

Submits an archive request to the DIVA Core Manager.
This call returns as soon as the Manager accepts the request. The application must call GET /requests/{requestid} to check that the operation was successful.

Parameters

Name	Description
Authorization string (header)	Bearer eyJhbGciOiJIUzUxMiJ9.eyJhdWQiOiI
body * required object (body)	Archive Request to create

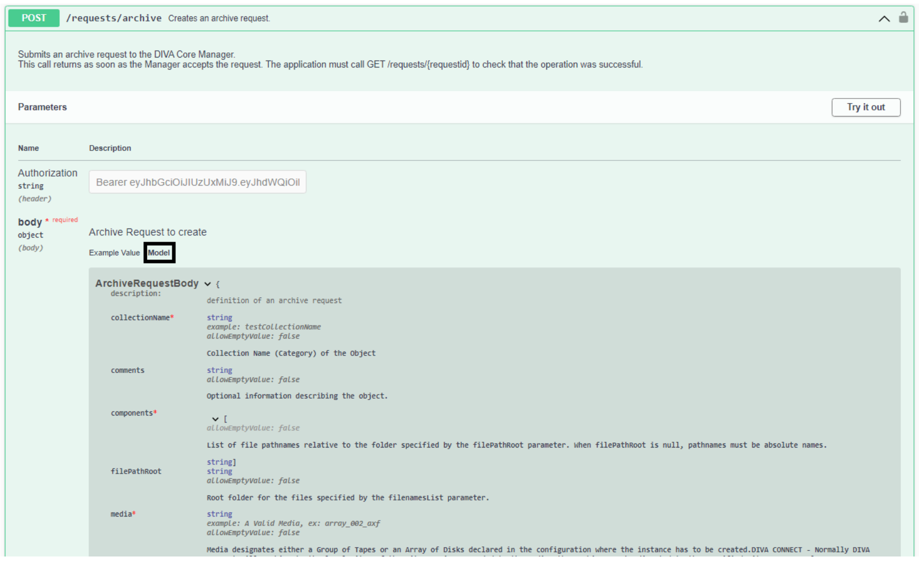
Edit Value | Model

```
{
  "collectionName": "testCollectionName",
  "comments": "string",
  "components": [
    "string"
  ],
  "filePathRoot": "string",
  "media": "A Valid Media, ex: array_002_axf",
  "objectName": "testObjectName",
  "options": "string",
  "priority": 50,
  "qos": 0,
  "sourceServer": "A Valid Source Server Name, ex: ftproot",
  "uuids": [
    "string"
  ]
}
```

Parameter content type: application/json

Specify the values and click Execute to submit the request.

Note: Click Model (next to the Example Value tab) to view a description of each field and a list possible values.



POST `/requests/archive` Creates an archive request.

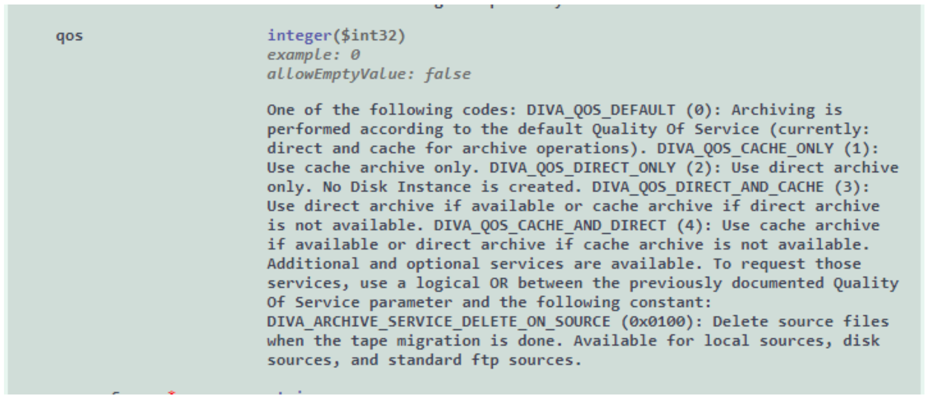
Submits an archive request to the DIVA Core Manager.
This call returns as soon as the Manager accepts the request. The application must call GET /requests/{requestid} to check that the operation was successful.

Parameters

Name	Description
Authorization string *(header)*	Bearer eyJhbGciOiJIUzUxMiJ9.eyJhdWQiOiI
body * required object *(body)*	Archive Request to create

```
ArchiveRequestBody {
   description:
                    definition of an archive request

   collectionName*  string
                    example: testCollectionName
                    allowEmptyValue: false

                    Collection Name (Category) of the Object

   comments         string
                    allowEmptyValue: false

                    Optional information describing the object.

   components*
                     [
                    allowEmptyValue: false

                    List of file pathnames relative to the folder specified by the filePathRoot parameter. When filePathRoot is null, pathnames must be absolute names.

                    string]
   filePathRoot     string
                    allowEmptyValue: false

                    Root folder for the files specified by the filenamesList parameter.

   media*           string
                    example: A Valid Media, ex: array_002_axf
                    allowEmptyValue: false

                    Media designates either a Group of Tapes or an Array of Disks declared in the configuration where the instance has to be created.DIVA CONNECT - Normally DIVA
```

For example, for qos, the list of possible QOS values and their meaning are displayed. A value of 2 signifies a QOS value of Direct-only.



```
qos                 integer($int32)
                    example: 0
                    allowEmptyValue: false

                    One of the following codes: DIVA_QOS_DEFAULT (0): Archiving is
                    performed according to the default Quality Of Service (currently:
                    direct and cache for archive operations). DIVA_QOS_CACHE_ONLY (1):
                    Use cache archive only. DIVA_QOS_DIRECT_ONLY (2): Use direct archive
                    only. No Disk Instance is created. DIVA_QOS_DIRECT_AND_CACHE (3):
                    Use direct archive if available or cache archive if direct archive
                    is not available. DIVA_QOS_CACHE_AND_DIRECT (4): Use cache archive
                    if available or direct archive if cache archive is not available.
                    Additional and optional services are available. To request those
                    services, use a logical OR between the previously documented Quality
                    Of Service parameter and the following constant:
                    DIVA_ARCHIVE_SERVICE_DELETE_ON_SOURCE (0x0100): Delete source files
                    when the tape migration is done. Available for local sources, disk
                    sources, and standard ftp sources.
```

Sample Python Program

Here is a sample program to obtain all Actors from DIVA using Python:

```
import requests

url = https://127.0.0.1:8765/dataservice/users/login

headers = {
        "Content-Type": "application/json; utf-8",
        "Accept": "application/json"
}
```

```
json = {
  "username": "enter_the_username_here",
  "password": "enter_the_password_here"
}

response = requests.post(url, headers=headers, json=json,
verify=False)

token = response.json()["token"]

print(token)

url = https://127.0.0.1:8765/manager/actors?page=1&size=5

headers = {
        "Accept": "application/json",
        "Authorization": token
}

response = requests.get(url, headers=headers, verify=False)

print(response.json())
```

Data Service API

Topics

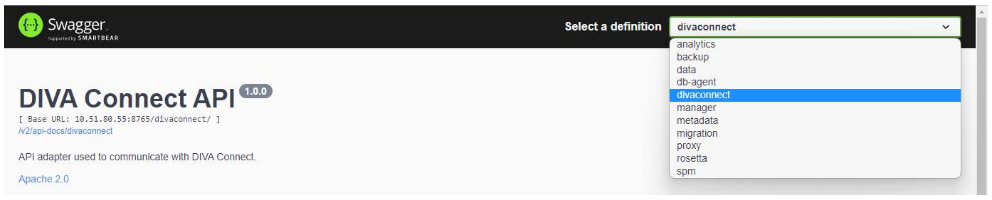
Overview

The REST API documentation is included in DIVA as HTTP documentation; which is accessible directly from within the REST API.The Swagger documentation for the REST API services is accessible using the following URL by replacing `localhost` with the correct IP address:

http://localhost:8765/webjars/swagger-ui/index.html

Important: Be sure to check the details in the Swagger API comments.

The Data Service endpoints can be switched to other endpoints using the menu at the top of the page:



DIVA Connect API 1.0.0

```
[ Base URL: 10.51.80.55:8765/divaconnect/ ]
```

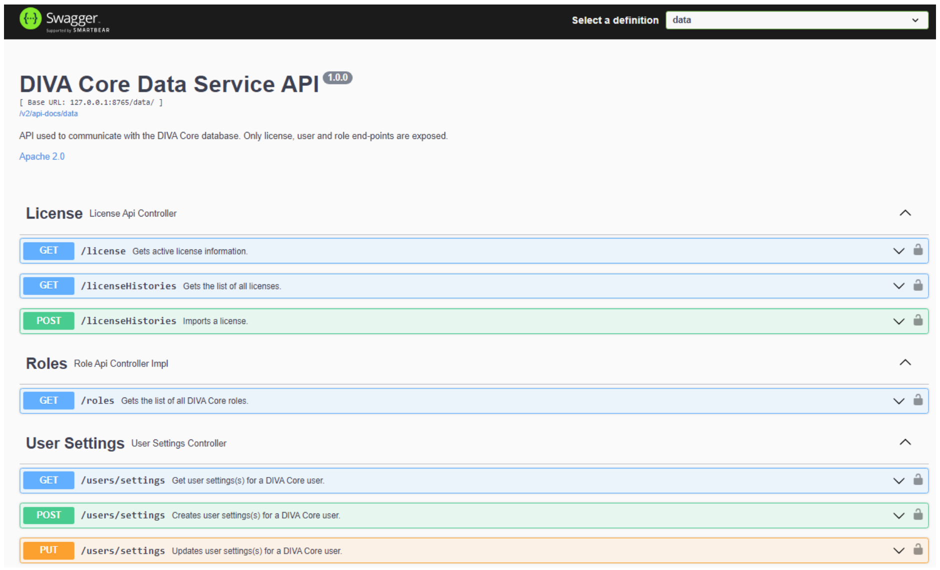
/v2/api-docs/divaconnect

API adapter used to communicate with DIVA Connect.

Apache 2.0

Data Service API

This is the API used to communicate with the database. Only user, profile, and endpoints are exposed. The Data Service is used to manage users, roles and profiles. After a user is created through *POST /users*, that user can obtain an access token through *POST /users/login* that will be needed for all future communication; including accessing all DIVA resources available in the Manager Endpoints.



DIVA Core Data Service API 1.0.0

```
[ Base URL: 127.0.0.1:8765/data/ ]
```

/v2/api-docs/data

API used to communicate with the DIVA Core database. Only license, user and role end-points are exposed.

Apache 2.0

License License Api Controller

GET `/license` Gets active license information.

GET `/licenseHistories` Gets the list of all licenses.

POST `/licenseHistories` Imports a license.

Roles Role Api Controller Impl

GET `/roles` Gets the list of all DIVA Core roles.

User Settings User Settings Controller

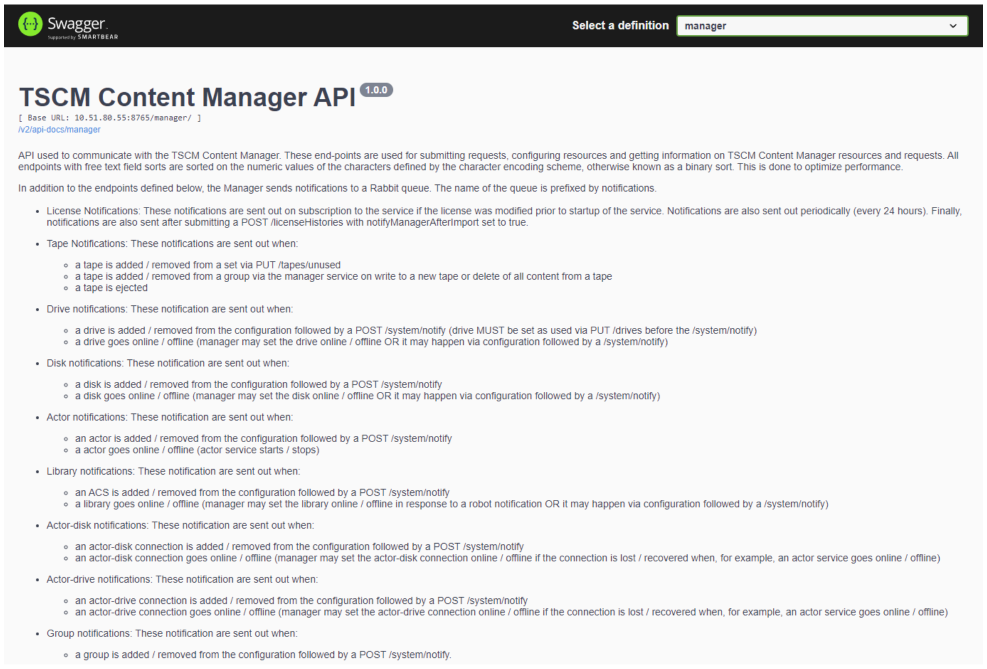
GET `/users/settings` Get user settings(s) for a DIVA Core user.

POST `/users/settings` Creates user settings(s) for a DIVA Core user.

PUT `/users/settings` Updates user settings(s) for a DIVA Core user.

DIVA Manager Endpoints

You use the API to communicate with DIVA. Use these endpoints to submit requests and obtain information on DIVA resources and requests.



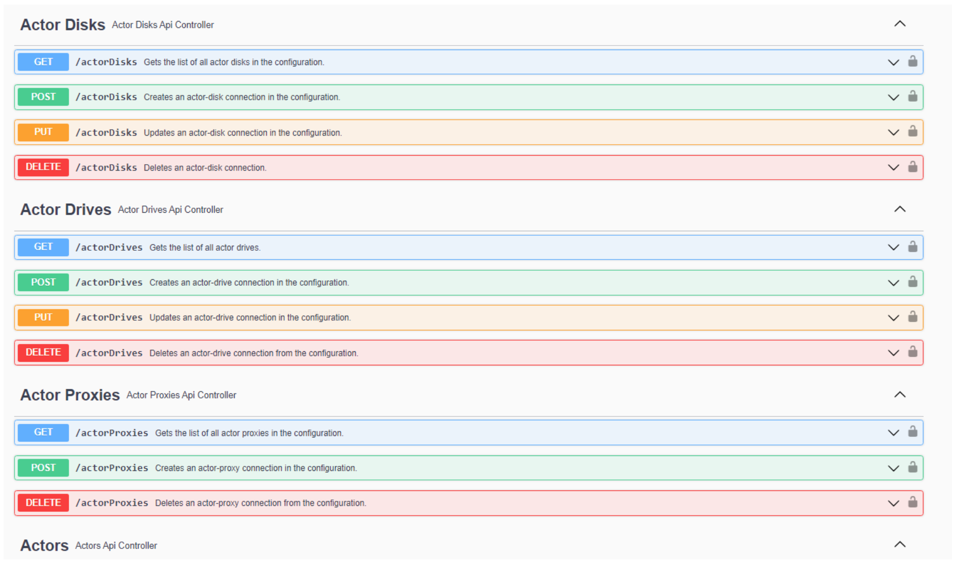
TSCM Content Manager API 1.0.0

[Base URL: 10.51.80.55:8765/manager/]
/v2/api-docs/manager

API used to communicate with the TSCM Content Manager. These end-points are used for submitting requests, configuring resources and getting information on TSCM Content Manager resources and requests. All endpoints with free text field sorts are sorted on the numeric values of the characters defined by the character encoding scheme, otherwise known as a binary sort. This is done to optimize performance.

In addition to the endpoints defined below, the Manager sends notifications to a Rabbit queue. The name of the queue is prefixed by notifications.

- License Notifications: These notifications are sent out on subscription to the service if the license was modified prior to startup of the service. Notifications are also sent out periodically (every 24 hours). Finally, notifications are also sent after submitting a POST /licenseHistories with notifyManagerAfterImport set to true.
- Tape Notifications: These notifications are sent out when:
  - a tape is added / removed from a set via PUT /tapes/unused
  - a tape is added / removed from a group via the manager service on write to a new tape or delete of all content from a tape
  - a tape is ejected
- Drive notifications: These notification are sent out when:
  - a drive is added / removed from the configuration followed by a POST /system/notify (drive MUST be set as used via PUT /drives before the /system/notify)
  - a drive goes online / offline (manager may set the drive online / offline OR it may happen via configuration followed by a /system/notify)
- Disk notifications: These notification are sent out when:
  - a disk is added / removed from the configuration followed by a POST /system/notify
  - a disk goes online / offline (manager may set the disk online / offline OR it may happen via configuration followed by a /system/notify)
- Actor notifications: These notification are sent out when:
  - an actor is added / removed from the configuration followed by a POST /system/notify
  - a actor goes online / offline (actor service starts / stops)
- Library notifications: These notification are sent out when:
  - an ACS is added / removed from the configuration followed by a POST /system/notify
  - a library goes online / offline (manager may set the library online / offline in response to a robot notification OR it may happen via configuration followed by a /system/notify)
- Actor-disk notifications: These notification are sent out when:
  - an actor-disk connection is added / removed from the configuration followed by a POST /system/notify
  - an actor-disk connection goes online / offline (manager may set the actor-disk connection online / offline if the connection is lost / recovered when, for example, an actor service goes online / offline)
- Actor-drive notifications: These notification are sent out when:
  - an actor-drive connection is added / removed from the configuration followed by a POST /system/notify
  - an actor-drive connection goes online / offline (manager may set the actor-drive connection online / offline if the connection is lost / recovered when, for example, an actor service goes online / offline)
- Group notifications: These notification are sent out when:
  - a group is added / removed from the configuration followed by a POST /system/notify.



Actor Disks Actor Disks Api Controller

GET	/actorDisks	Gets the list of all actor disks in the configuration.
POST	/actorDisks	Creates an actor-disk connection in the configuration.
PUT	/actorDisks	Updates an actor-disk connection in the configuration.
DELETE	/actorDisks	Deletes an actor-disk connection.

Actor Drives Actor Drives Api Controller

GET	/actorDrives	Gets the list of all actor drives.
POST	/actorDrives	Creates an actor-drive connection in the configuration.
PUT	/actorDrives	Updates an actor-drive connection in the configuration.
DELETE	/actorDrives	Deletes an actor-drive connection from the configuration.

Actor Proxies Actor Proxies Api Controller

GET	/actorProxies	Gets the list of all actor proxies in the configuration.
POST	/actorProxies	Creates an actor-proxy connection in the configuration.
DELETE	/actorProxies	Deletes an actor-proxy connection from the configuration.

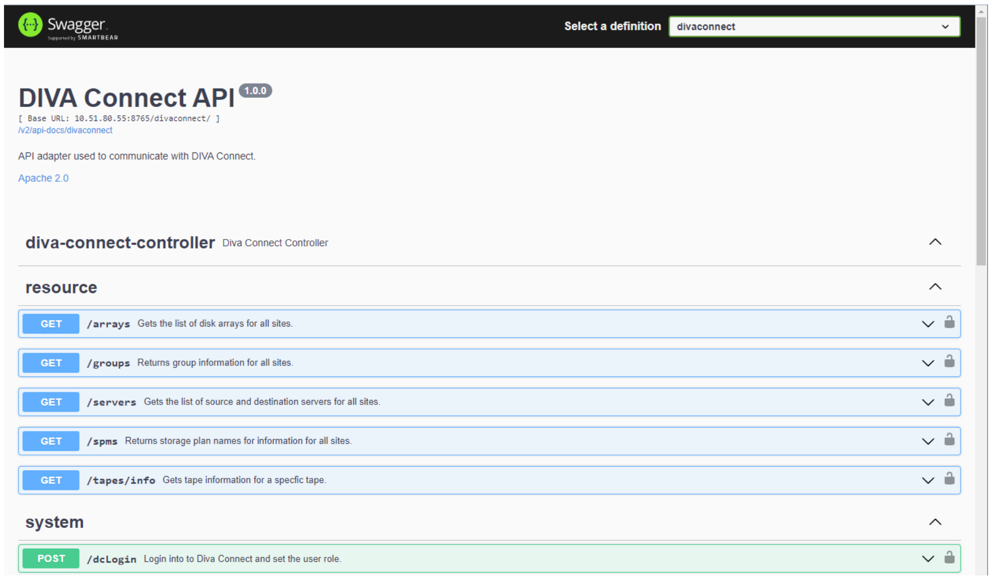
Actors Actors Api Controller

DIVA Connect REST API

The REST API can be used to send requests and commands to DIVA Connect. The same port used for the Client Web Connections is used for this API. The REST API supports only MultiDIVA Mode.

See the DIVA Connect documentation on the DIVA Support Portal at https://www.telestream.net/telestream-support/ for detailed information.

Swagger contains DIVA Connect REST API endpoints as shown here:



DIVA Connect API 1.0.0

[Base URL: 10.51.80.55:8765/divaconnect/]
/v2/api-docs/divaconnect

API adapter used to communicate with DIVA Connect.

Apache 2.0

diva-connect-controller Diva Connect Controller

resource

GET /arrays Gets the list of disk arrays for all sites.

GET /groups Returns group information for all sites.

GET /servers Gets the list of source and destination servers for all sites.

GET /spms Returns storage plan names for information for all sites.

GET /tapes/info Gets tape information for a specfic tape.

system

POST /dcLogin Login into to Diva Connect and set the user role.

Workflows

This chapter describes the DIVA API and Authentication Token Workflows. The REST API uses JWT (JSON Web Token) authentication specified in the authorization header of all requests. To obtain the token, *POST* to */users/login* on the data service; passing in the user name and password. There is a specific endpoint to get a authentication token and all the functions of the REST API require this token to function properly.

Authentication Token Workflow

The authentication phase is mandatory in order to get a token that will be used for any following API call. A token is configurable and valid for 1440 minutes (24 hours) by default. The maximum is 10080 minutes (7 days). It is advised to authenticate one time at the start of your application before the 1st call to a DIVA API call, and then use that token as long as it is valid. Any HTTP request using an invalid or expired token will fail with HTTP error code 403 (access denied).

The following process is the authentication workflow.

1. Upon login the user will receive an authentication token.
2. An access token must be used to access secured endpoints. It will automatically expire after one day. Alternatively, a user may delete an access token by calling */users/logout*.
3. When an access token expires or is deleted, the client is considered as logged out and must login again.

Roles

A user may belong to one of five roles; sysadmin, admin, advoperator, operator, or user.

A user may perform all basic GET operations including the following:

- `POST /users/login`
- `POST /users/logout`
- `PUT /users/{userName}/password`
- `GET /profile`

- `PUT /profile`
- `GET /users`
- `GET /roles`
- `GET ANY RESOURCE (for example, GET /actors)`

An Operator may perform all the operations of a user and the following additional operations:

- `POST /requests/archive`
- `POST /requests/restore`
- `POST /requests/copy`

An Advanced Operator (advoperator) may perform all the operations of an operator and the following additional operations:

- `PUT /requests`
- `POST /requests/transferFiles`
- `POST /requests/insertTape`
- `POST /requests/ejectTape`
- `POST /requests/repackTape`
- `POST /requests/exportTape`
- `POST /requests/importTape`

An Administrator (admin) may perform all operations of an advoperator and the following additional operations:

- `POST /requests/delete`
- `POST /requests/serverDelete`

A System Administrator (sysadmin) may perform all operations of an administrator and the following additional operations:

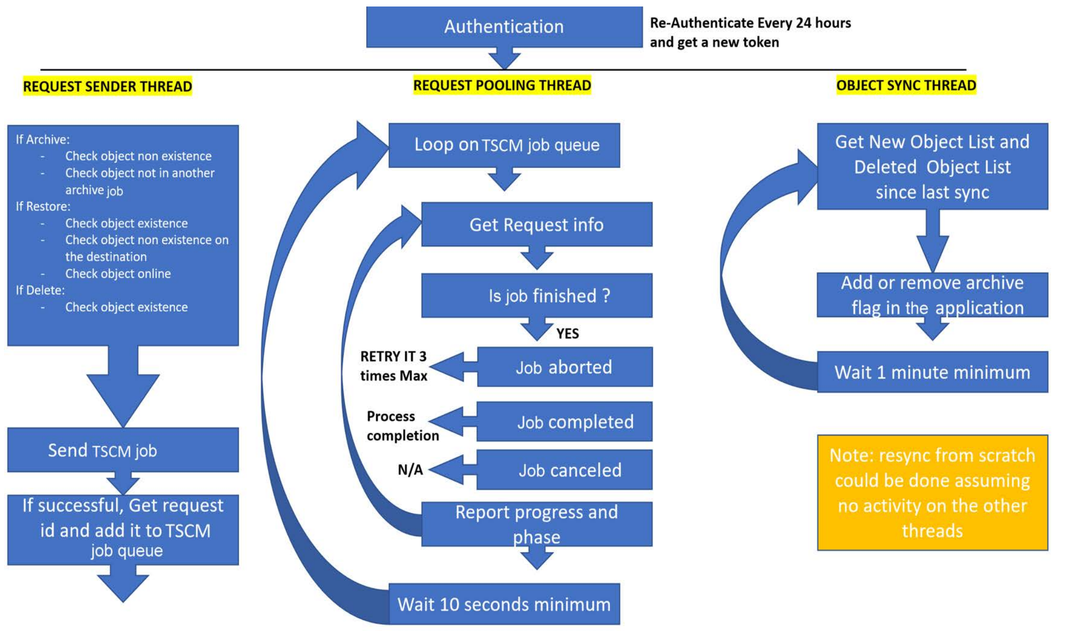
- `POST /users`
- `DELETE /users`
- `GET /users`
- `GET /roles`

DIVA API Workflows

The following guidelines should be used to develop workflows using the DIVA API:

- First authentication: if possible use only one authentication to DIVA at the start of the application and use the token returned for further API calls. Do not authenticate multiple times, and in particular not before each DIVA request.
- Send the DIVA request (archive, restore, and so on) using the token from the last step and get the Request ID. Add the Request ID to the DIVA request queue.
- Pool every `n` seconds on the DIVA request queue list using `getRequestInfo`. Wait a minimum of 10 seconds between each pooling phase.
- The progress and phase can be obtained for each running request.
- Any running request can be canceled.
- A finished request can be removed from the DIVA request queue. A finished request will be COMPLETED, PARTIALLY_COMPLETED, ABORTED, or CANCELLED.
- Avoid retrying too many times if a request fails.
- Before restoring an object, use `divaGetObjectInfo` to know if the object is online; there is no need to try to restore an offline object because it will fail.
- Try to develop a sync (or resync) mechanism to sync the application with DIVA objects using the Since Date option to discover new and deleted objects.

After authenticated, three different threads could be created to manage the DIVA workflows as shown in the following figure:


Authentication
Re-Authenticate Every 24 hours and get a new token
REQUEST SENDER THREAD
REQUEST POOLING THREAD
OBJECT SYNC THREAD
If Archive:
- Check object non existence
- Check object not in another archive job
If Restore:
- Check object existence
- Check object non existence on the destination
- Check object online
If Delete:
- Check object existence
Send TSCM job
If successful, Get request id and add it to TSCM job queue
Loop on TSCM job queue
Get Request info
Is job finished ?
YES
RETRY IT 3 times Max
Job aborted
Process completion
Job completed
N/A
Job canceled
Report progress and phase
Wait 10 seconds minimum
Get New Object List and Deleted Object List since last sync
Add or remove archive flag in the application
Wait 1 minute minimum
Note: resync from scratch could be done assuming no activity on the other threads

DIVA Request Status Codes

This table identifies DIVA request status codes:

Code	Name	Description
1000	DIVA_OK	Success
1001	DIVA_ERR_UNKNOWN	Error: unknown error
1002	DIVA_ERR_INTERNAL	Error: internal error
1003	DIVA_ERR_NO_ARCHIVE_SYSTEM	Error: no archive system
1004	DIVA_ERR_BROKEN_CONNECTION	Error: broken connection
1005	DIVA_ERR_DISCONNECTING	Error: while disconnecting
1006	DIVA_ERR_ALREADY_CONNECTED	Error: already connected
1007	DIVA_ERR_WRONG_VERSION	Error: wrong software version
1008	DIVA_ERR_INVALID_PARAMETER	Error: invalid parameter
1009	DIVA_ERR_OBJECT_DOESNT_EXIST	Error: Object doesn't exist
1010	DIVA_ERR_SEVERAL_OBJECTS	Error: several objects with this name
1011	DIVA_ERR_NO_SUCH_REQUEST	Error: no such request
1012	DIVA_ERR_NOT_CANCELABLE	Error: request is not cancelable
1013	DIVA_ERR_SYSTEM_IDLE	Error: DIVA is idle
1014	DIVA_ERR_WRONG_LIST_SIZE	Error: wrong objects list size
1015	DIVA_ERR_LIST_NOT_INITIALIZED	Error: Objects list is not initialized
1016	DIVA_ERR_OBJECT_ALREADY_EXISTS	Error: Object already exists
1017	DIVA_ERR_GROUP_DOESNT_EXIST	Error: Tape Group, media or storage plan does not exist
1018	DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST	Error: source or destination doesn't exist
1019	DIVA_WARN_NO_MORE_OBJECTS	Warning: no more objects
1020	DIVA_ERR_NOT_CONNECTED	Error: not connected
1021	DIVA_ERR_GROUP_ALREADY_EXISTS	Error: Tape Group, media or storage plan already exists
1022	DIVA_ERR_GROUP_IN_USE	Error: archived objects belong to this Tape Group

Code	Name	Description
1023	DIVA_ERR_OBJECT_OFFLINE	Error: Object offline
1024	DIVA_ERR_TIMEOUT	Error: timeout
1025	DIVA_ERR_LAST_INSTANCE	Error: last instance
1026	DIVA_ERR_PATH_DESTINATION	Error: destination path must be complete
1027	DIVA_ERR_INSTANCE_DOESNT_EXIST	Error: instance does not exist
1028	DIVA_ERR_INSTANCE_OFFLINE	Error: instance offline
1029	DIVA_ERR_INSTANCE_MUST_BE_ON_TAPE	Error: instance must be on tape
1030	DIVA_ERR_NO_INSTANCE_TAPE_EXIST	Error: no tape instance exists
1031	DIVA_ERR_OBJECT_IN_USE	Error: Object in use
1032	DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS	Error: cannot accept more requests
1033	DIVA_ERR_TAPE_DOESNT_EXIST	Error: tape doesn't exist
1034	DIVA_ERR_INVALID_INSTANCE_TYPE	Error: invalid instance type
1035	DIVA_ERR_ACCESS_DENIED	Error: access denied
1036	DIVA_ERR_OBJECT_PARTIALLY_DELETED	Error: Object is partially deleted
1037	DIVA_ERR_LICENSE_DOES_NOT_SUPPORT_THIS_FEATURE	License does not support this feature
1038	DIVA_ERR_COMPONENT_NOT_FOUND	Error: component not found
1039	DIVA_ERR_OBJECT_IS_LOCKED	Error: Object is locked
1040	DIVA_ERR_OBJECT_BEING_ARCHIVED	Error: Object is being archived

This table identifies possible status codes for unsuccessful Archive requests:

Code	Name	Description
1002	DIVA_ERR_INTERNAL	Error: internal error
1008	DIVA_ERR_INVALID_PARAMETER	Error: invalid parameter
1016	DIVA_ERR_OBJECT_ALREADY_EXISTS	Error: Object already exists
1018	DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST	Error: source or destination doesn't exist
1040	DIVA_ERR_OBJECT_BEING_ARCHIVED	Error: Object is being archived

Partial Restore Request Formats and Manager Responses

The following formats—each identified by an INT value in *format*—are used when issuing requests to DIVA Manager:

- 0—Bytes (range)
- 1—Not Used
- 2—Video GXF (timecode)
- 3—Video SEA (timecode)
- 4—Video AVI MATROX (timecode)
- 5—Video MPEG2 TS (timecode)
- 6—Video MXF (timecode)
- 7—Video Pinnacle (timecode)
- 8—Video Omneon (timecode)
- 9—Video Leitch (timecode)
- 10—Video Quantel (timecode)
- 11—Autodetect which video format (timecode)
- 12—File/Folder Based
- 13—DPX (range)

Request and Response Sample

Here are Partial Restore requests and Manager responses. Take note of the differences in offsets and formats.

Sample 1: Body for Bytes Partial Restore

```
{
  "destinationServer": "sourcedest",
  "minRequestPriority": -1,
  "instance": -1,
  "qos": 0,
  "offsets": [
    {
    "destinationFile": "DNxHD-mxf-wrap-conf.mov",
    "offsetPairs": [
      {
        "bytesEnd": 1,
        "bytesBegin": 0,
        "timeCode": false
      },
      {
        "bytesEnd": 2,
        "bytesBegin": 1,
        "timeCode": false
      }
    ],
    "sourceFile": "DNxHD-mxf-wrap-conf.mov"
    }
  ],
  "format": 0,
  "options": " ",
  "objectName": "Partial File",
  "maxRequestPriority": 100,
  "priority": -1,
  "filePathRoot": "restore",
  "collectionName": "Restore All Basic PFR",
  "destinationServer": "sourcedest",
  "minRequestPriority": -1,
  "instance": -1,
  "qos": 0,
  "offsets": [
    {
"offsets": [
  {
    "destinationFile": "DNxHD-mxf-wrap-conf.mov",
    "offsetPairs": [
      {
        "bytesEnd": 1,
        "bytesBegin": 0,
          "timeCode": false
      },
      {
        "bytesEnd": 2,
        "bytesBegin": 1,
        "timeCode": false
      }
    ],
    "sourceFile": "DNxHD-mxf-wrap-conf.mov"
  }
],
  "format": 0,
  "options": " ",
  "objectName": "Partial File",
  "maxRequestPriority": 100,
  "priority": -1,
  "filePathRoot": "restore",
  "collectionName": "Restore All Basic PFR"
  }
}
```

Sample 2: Body for Video GXF (timecode) Partial Restore

```
  {
    "destinationServer": "sourcedest",
    "minRequestPriority": -1,
    "instance": -1,
    "qos": 0,
    "offsets": [
      {
        "destinationFile": "Profile.gxf",
        "offsetPairs": [
          {
            "timeCodeBegin": "00:00:00:00",
            "timeCodeEnd": "00:00:00:01",
            "bytesEnd": 0,
            "bytesBegin": 0,
            "timeCode": true
          }
        ],
        "sourceFile": "Profile.gxf"
      }
    ],
    "format": 2,
    "options": " ",
    "objectName": "Partial File",
    "maxRequestPriority": 100,
    "priority": -1,
    "filePathRoot": "restore",
    "collectionName": "Restore All Basic PFR"
  }
```

Sample 3: Body for File-Folder based Partial Restore

```
  {
    "destinationServer": "sourcedest",
    "minRequestPriority": -1,
    "instance": -1,
    "qos": 0,
    "offsets": [
      {
        "fileFolder": {
          "name": "DNxHD_mxf_wrap_conf.mov",
          "option": ""
        }
      },
      {
        "fileFolder": {
          "name": "test.mov",
          "option": ""
        }
      }
    ],
    "format": 12,
    "options": " ",
    "objectName": "Partial File",
    "maxRequestPriority": 100,
    "priority": -1,
    "filePathRoot": "restore",
    "collectionName": "Restore All Basic PFR"
  }
```

Sample 4: Body for DPX (Range) PR

```
  {
    "destinationServer": "sourcedest",
    "minRequestPriority": -1,
    "instance": -1,
    "qos": 0,
    "offsets": [
      {
        "range": {
          "end": 2,
          "begin": 1
        }
      },
      {
        "range": {
          "end": 4,
          "begin": 3
        }
      }
    ],
    "format": 13,
    "options": " ",
    "objectName": "Partial File",
    "maxRequestPriority": 100,
    "priority": -1,
    "filePathRoot": "restore",
    "collectionName": "Restore All Basic PFR"
  }
```