Telestream

DIVA Core

REST API

Programmer’s Guide

Release: 8.2

Revision: 1.2

September 26, 2022

Copyrights and Trademark Notices

Contents

Telestream Contact Information 4

Overview 5

New Terminology 6
DIVA Core Concepts 7
    Archive Request 7
    Restore Request 8
    Partial Restore 8
    Delete Request 8
Main DIVA Core API Calls 9

Data Service API 10

Data Service API 11
Switching to Core Manager Endpoints 11

Workflows 13

Authentication Token Workflow 14
Roles 15
DIVA Core API Workflows 17
DIVA Core Request Status Codes 18
Partial Restore Request Formats and Core Manager Responses 20
    Request and Response Samples 20
    Sample 1: Body for Bytes Partial Restore 20
    Sample 2: Body for Video GXF (timecode) Partial Restore 20
    Sample 3: Body for File-Folder based Partial Restore 20
    Sample 4: Body for DPX (Range) PR 20

Getting Started 21

Initial Configuration 22
Sample Program 28

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 Core Technical Support	Web Site: https://www.telestream.net/telestream-support/diva/support.htm 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.

Overview

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

DIVA Core exposes its functionality through a REST interface. It is self-contained in DIVA Core 8.0 and all future DIVA Core releases. In the 8.0 release, the API is used exclusively by the DIVA Core Web Application.

Notes: Telestream recommends using the REST API rather than the previous existing APIs (that is, DIVA Core Enterprise Connect, DIVAS, Java and C++). Although all previous APIs will remain available, the REST API offers new and enhance features.

JSON files can be downloaded for the REST API from SharePoint here: https://tinyurl.com/y5c36jeb

---

Topics:

■New Terminology
■DIVA Core Concepts
■Main DIVA Core API Calls

New Terminology

The following terminology has been updated to reflect standardization efforts across all DIVA and Kumulate applications. There will be some variations in the documentation compared to the interface until everything is switched over to the new terminology; the documentation uses the new terms wherever possible.

- Running Requests are now called Jobs
- Request History is now called Job History
- Libraries are now called Managed Storage
- Datahub is now called Actor
- Proxyhub is now called Proxy Actor
- DIVA Core and DIVA Manager are now called DIVA Core / Core / Core Manager
- Category is now called Collection
- Source/Destination is now called Unmanaged Storage Repository
- Storage Repository is now called Managed Storage Repository
- Object is now called Virtual Object
- Group is now called Tape Group
- Link is now called Storage Link
- Storage Plan Manager is now called Storage Policy Manager
- Drop Folder Monitor (DFM) is now called Watch Folder Monitor (WFM)
- DIVA Command and Control Panel are now called System Management App
- DIVA Analytics and DIVAProtect are now called Analytics App

DIVA Core Concepts

The following information are standard DIVA Core concepts.

Archive Request

DIVA Core stores Virtual Objects; a Virtual Object is a set of files referring to an asset or a clip. a Virtual 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.

a Virtual Object is identified by a name and a Collection. You can choose whatever names for Virtual Object name and Collection you want, DIVA Core only checks that the Virtual Object name + Collection combination is unique.

In DIVA Core, a Collection is like a name extension and should not be confused with a Tape Group. You can put any name as the Collection. Telestream recommends putting your application or company name so we can identify who has sent a request. Should you use the same Virtual Object name for different clips (typically hires and lowers), you can put a different Collection to distinguish those clips.

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

Media Name is the DIVA Core device used for storing the Virtual 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). You can get the list of all Arrays and Tape Groups from DIVA Core but you do not necessarily need to expose all of them to the end user. The Media can be also a Storage Plan. You should check with the customer and the DIVA Core Project Manager about which Media to expose to the end user.

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

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

Note: The Source list can be obtained using the *GET /servers* DIVA Core 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 you can specify a value.

If the Delete From Source option is 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:

- Virtual Object Name
- Virtual Object Collection
- Unmanaged Storage Repository Server Name
- The File Path Root; if empty, DIVA Core will take the File Path Root used during the Archive request and will overwrite the Virtual 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 Virtual Object Request will delete all copies of that Virtual Object whether they are on disk, tape (in the tape library or external), or in the cloud. You must specify the Virtual Object Name and Virtual Object Collection.

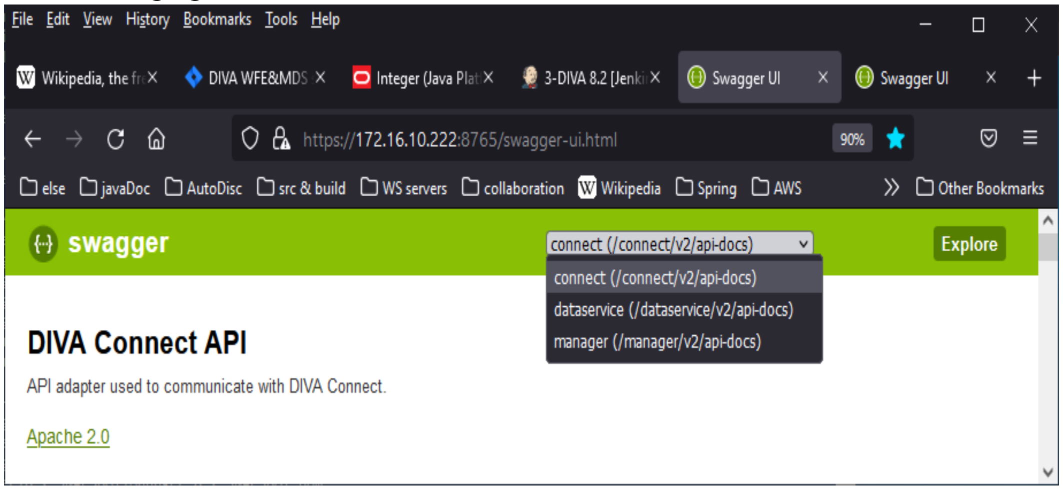
Main DIVA Core API Calls

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

- */users/login Post*
- */users/logout Post*
- */groups Get*
- */arrays Get*
- */object/info Get*
- */objects/list Get*
- */requests Get*
- */requests/archive Post*
- */requests/cancel Post*
- */requests/delete Post*
- */requests/partialRestore Post*
- */requests/restore Post*
- */requests/{requestId} Get*
- */versions Get*

Data Service API

The REST API detailed documentation is included in DIVA Core as HTTP documentation; which is accessible directly from within the REST API.The Swagger documentation for the REST API services is accessible at https://localhost:8765/api-docs. The Swagger documentation may also contain DIVA Connect REST API documentation as shown in the following figure:


File Edit View History Bookmarks Tools Help
Wikipedia, the fr
DIVA WFE&MDS
Integer (Java Plat
3-DIVA 8.2 [Jenki
Swagger UI
Swagger UI
https://172.16.10.222:8765/swagger-ui.html
90%
else javaDoc AutoDisc src & build WS servers collaboration Wikipedia Spring AWS Other Bookmarks
swagger
connect (/connect/v2/api-docs)
Explore
connect (/connect/v2/api-docs)
dataservice (/dataservice/v2/api-docs)
manager (/manager/v2/api-docs)

DIVA Connect API

API adapter used to communicate with DIVA Connect.

Apache 2.0

You can switch from the Data Service endpoints to the Core Manager Service endpoints using the pull down menu at the top of the page.

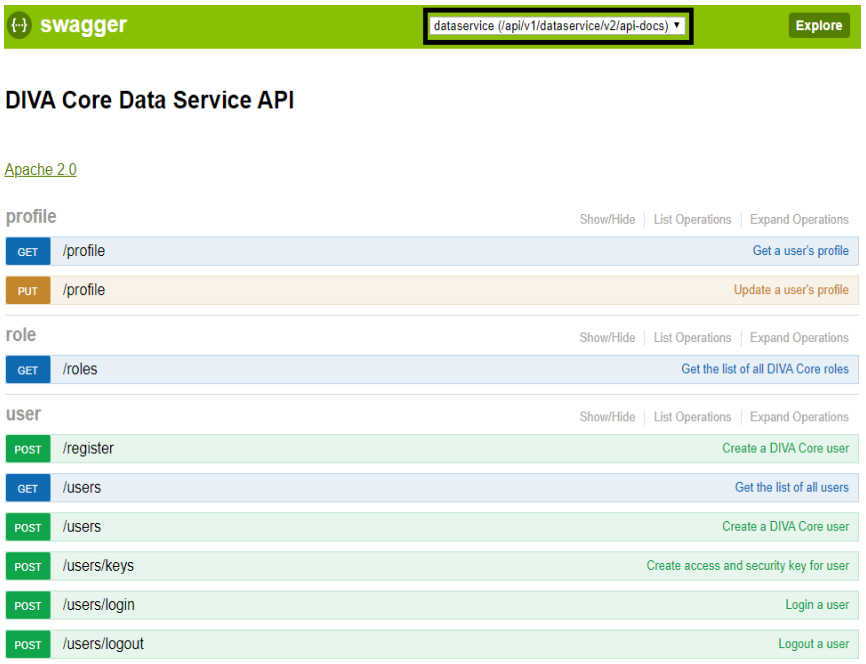
Topics:

■Data Service API
■Switching to Core Manager Endpoints

Data Service API

This is the API used to communicate with the Core 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 Core resources available in the Core Manager Endpoints.

Switching to Core Manager Endpoints



DIVA Core Data Service API

Apache 2.0

profile — Show/Hide | List Operations | Expand Operations

GET	/profile	Get a user's profile
PUT	/profile	Update a user's profile

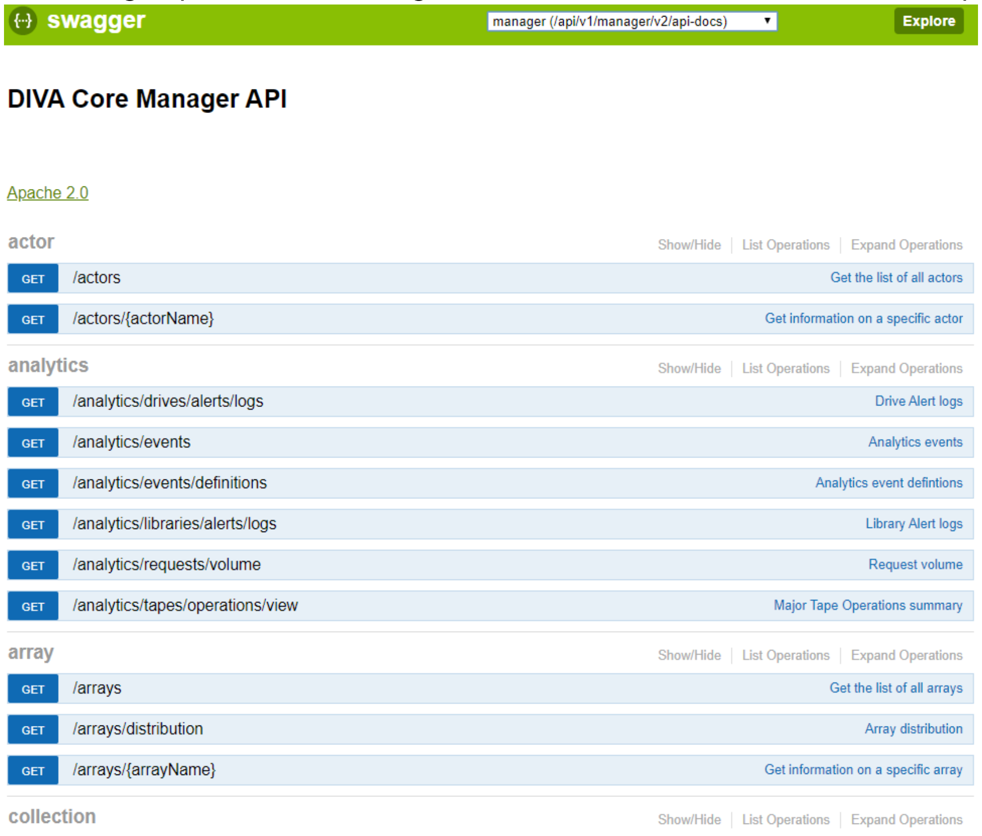
role — Show/Hide | List Operations | Expand Operations

GET	/roles	Get the list of all DIVA Core roles

user — Show/Hide | List Operations | Expand Operations

POST	/register	Create a DIVA Core user
GET	/users	Get the list of all users
POST	/users	Create a DIVA Core user
POST	/users/keys	Create access and security key for user
POST	/users/login	Login a user
POST	/users/logout	Logout a user

The API is used to communicate with the Core Manager. These endpoints are used for submitting requests and obtaining information on DIVA Core resources and requests.



DIVA Core Manager API

Apache 2.0

actor — Show/Hide | List Operations | Expand Operations

GET	/actors	Get the list of all actors
GET	/actors/{actorName}	Get information on a specific actor

analytics — Show/Hide | List Operations | Expand Operations

GET	/analytics/drives/alerts/logs	Drive Alert logs
GET	/analytics/events	Analytics events
GET	/analytics/events/definitions	Analytics event defintions
GET	/analytics/libraries/alerts/logs	Library Alert logs
GET	/analytics/requests/volume	Request volume
GET	/analytics/tapes/operations/view	Major Tape Operations summary

array — Show/Hide | List Operations | Expand Operations

GET	/arrays	Get the list of all arrays
GET	/arrays/distribution	Array distribution
GET	/arrays/{arrayName}	Get information on a specific array

collection — Show/Hide | List Operations | Expand Operations

Workflows

This chapter describes the DIVA Core 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, you must *POST* to */users/login* on the data service; passing in your 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.

Topics:

- Authentication Token Workflow
- Roles
- DIVA Core API Workflows
- DIVA Core Request Status Codes
- Partial Restore Request Formats and Core Manager Responses

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 valid 24 hours. It is advised to authenticate one time at the start of your application before the 1st call to a DIVA Core 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 log in 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:

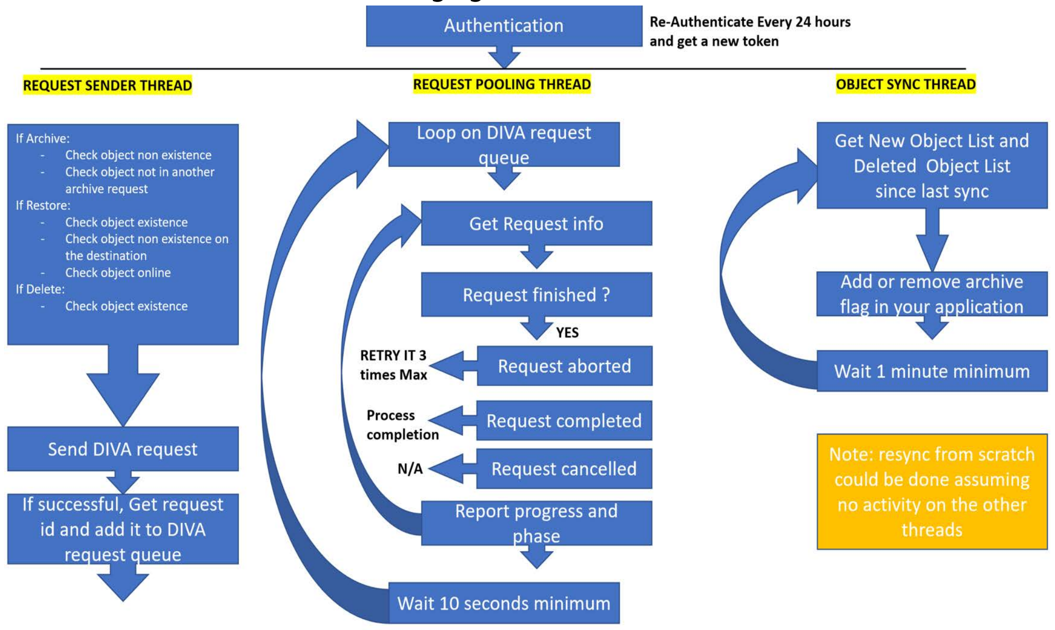
- *POST /users*
- *DELETE /users*
- *GET /users*
- *GET /roles*

DIVA Core API Workflows

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

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

After authenticated, three different threads could be created to manage the DIVA Core 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 request
If Restore:
- Check object existence
- Check object non existence on the destination
- Check object online
If Delete:
- Check object existence
Send DIVA request
If successful, Get request id and add it to DIVA request queue
Loop on DIVA request queue
Get Request info
Request finished ?
YES
RETRY IT 3 times Max
Request aborted
Process completion
Request completed
N/A
Request cancelled
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 your application
Wait 1 minute minimum
Note: resync from scratch could be done assuming no activity on the other threads

DIVA Core Request Status Codes

The following table identifies DIVA Core 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: Virtual Object doesn't exist
1010	DIVA_ERR_SEVERAL_OBJECTS	Error: several Virtual Objects with this name
1011	DIVA_ERR_NO_SUCH_REQUEST	Error: no such request
1012	DIVA_ERR_NOT_CANCELABLE	Error: request is not cancellable
1013	DIVA_ERR_SYSTEM_IDLE	Error: DIVA system is idle
1014	DIVA_ERR_WRONG_LIST_SIZE	Error: wrong Virtual Objects list size
1015	DIVA_ERR_LIST_NOT_INITIALIZED	Error: Virtual Objects list is not initialized
1016	DIVA_ERR_OBJECT_ALREADY_EXISTS	Error: Virtual 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 Virtual Objects
1020	DIVA_ERR_NOT_CONNECTED	Error: not connected

Code	Name	Description
1021	DIVA_ERR_GROUP_ALREADY_EXISTS	Error: Tape Group, media or storage plan already exists
1022	DIVA_ERR_GROUP_IN_USE	Error: archived Virtual Objects belong to this Tape Group
1023	DIVA_ERR_OBJECT_OFFLINE	Error: Virtual 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: Virtual 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: Virtual 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: Virtual Object is locked
1040	DIVA_ERR_OBJECT_BEING_ARCHIVED	Error: Virtual Object is being archived

The following 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: Virtual 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: Virtual Object is being archived

Partial Restore Request Formats and Core Manager Responses

The following formats are used when issuing requests to the Core 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 Samples

The following subsections are sample Partial Restore requests and Core Manager responses. Take note of the differences in offsets and formats.

Sample 1: Body for Bytes Partial Restore

```
{
"destinationServer": "sourcedest",
"minRequestPriority": -1,
"instance": -1,
```

Getting Started

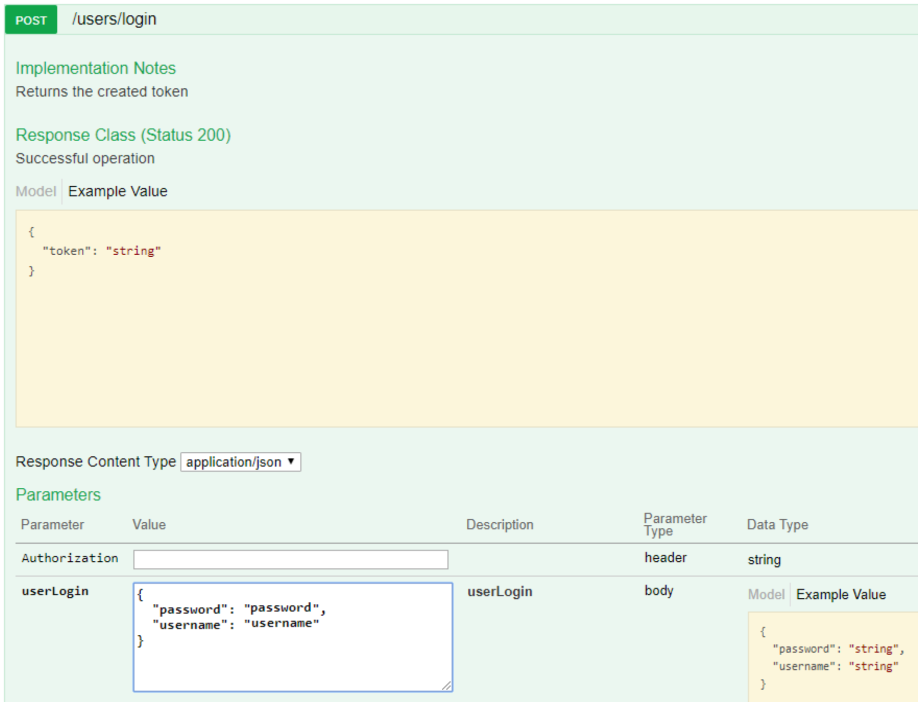
This chapter guides the user through getting started using the DIVA Core REST API.

Topics:

- Initial Configuration
- Sample Program

Initial Configuration

During installation a user will be created by either the Telestream Installer, or your DIVA Core Administrator. You must obtain this information from the person who created the user; all automations and API calls will use that login and password combination. Go to the *POST users/login* endpoint and specify the login and password to log in; this is sufficient to get a token and proceed with the rest of the API calls.


POST /users/login

Implementation Notes
Returns the created token

Response Class (Status 200)
Successful operation

Model | Example Value

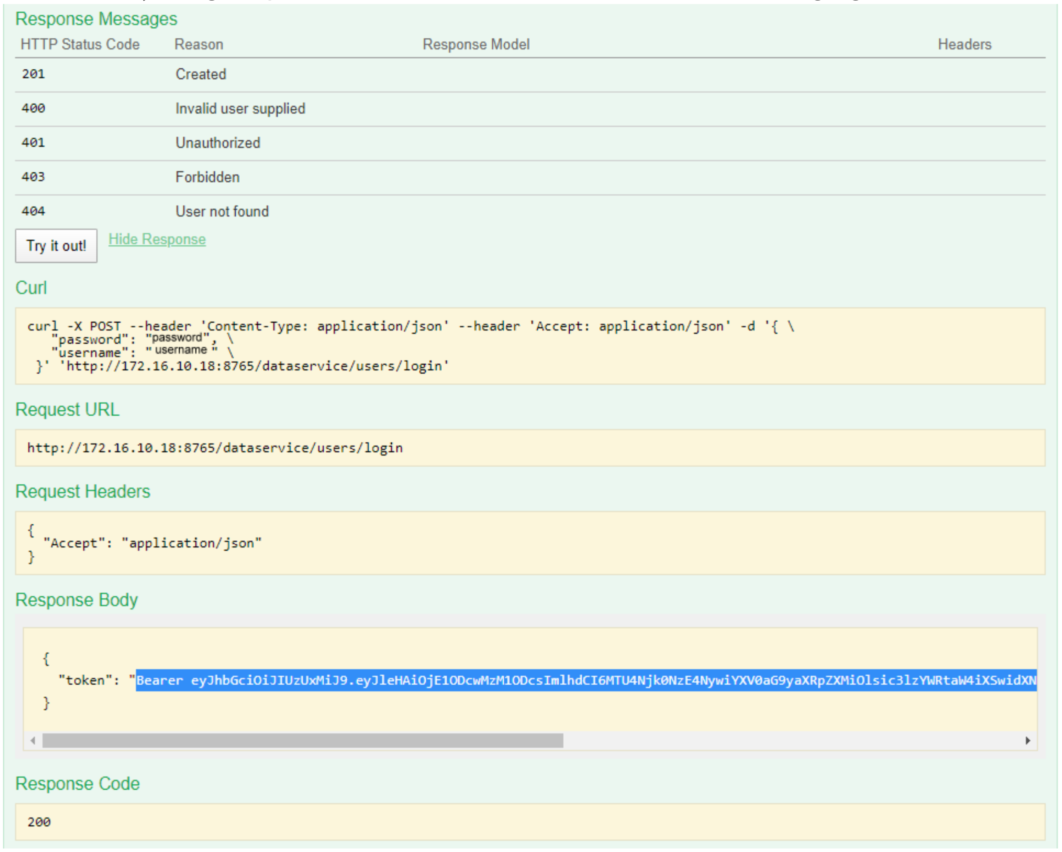
```
{
  "token": "string"
}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization			header	string
userLogin	`{ "password": "password", "username": "username" }`	userLogin	body	Model | Example Value `{ "password": "string", "username": "string" }`

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:



Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
400	Invalid user supplied		
401	Unauthorized		
403	Forbidden		
404	User not found		

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
   "password": "password", \
   "username": " username " \
 }' 'http://172.16.10.18:8765/dataservice/users/login'
```

Request URL

```
http://172.16.10.18:8765/dataservice/users/login
```

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
  "token": "Bearer eyJhbGciOiJIUzUxMiJ9.eyJleHAiOjE1ODcwMzM1ODcsImlhdCI6MTU4Njk0NzE4NywiYXV0aG9yaXRpZXMiOlsic3lzYWRtaW4iXSwidXN
}
```

Response Code

```
200
```

You must submit a *POST /users* request by entering the token in the Authorization field to create a user. You must also specify the user name, password and role of the user you will create (see the following figure).

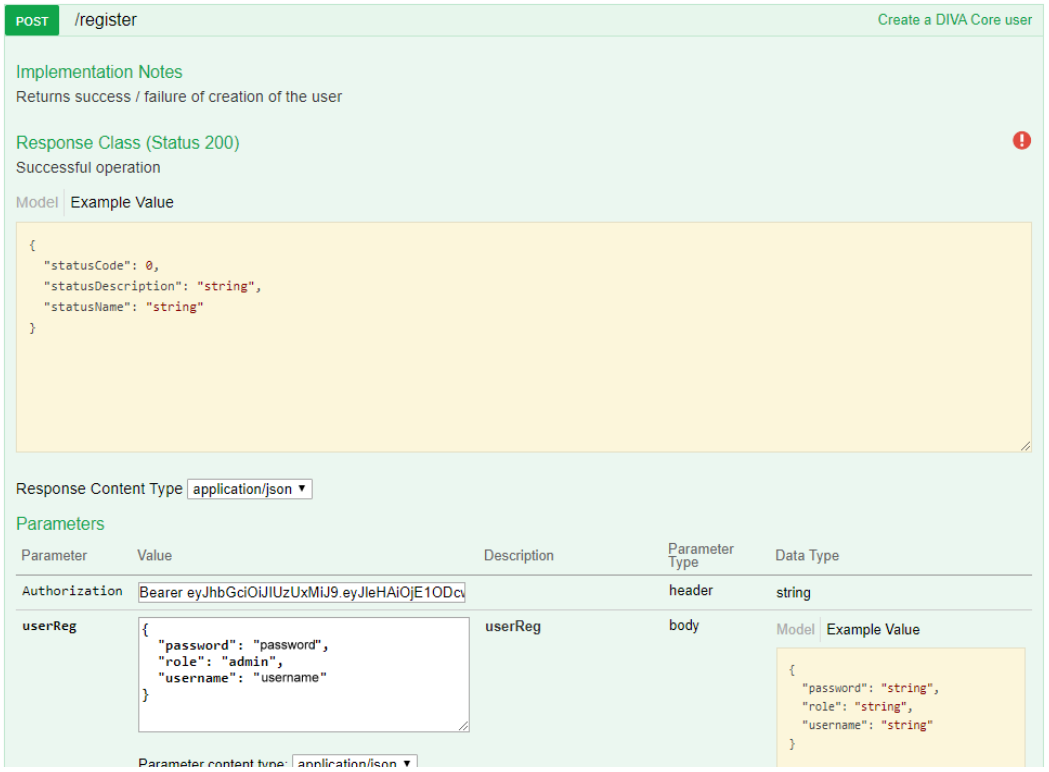
Note: Call *GET /roles* to get a list of the possible roles.

All DIVA Core 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.



POST /register — Create a DIVA Core user

Implementation Notes
Returns success / failure of creation of the user

Response Class (Status 200)
Successful operation

Model | Example Value

```
{
  "statusCode": 0,
  "statusDescription": "string",
  "statusName": "string"
}
```

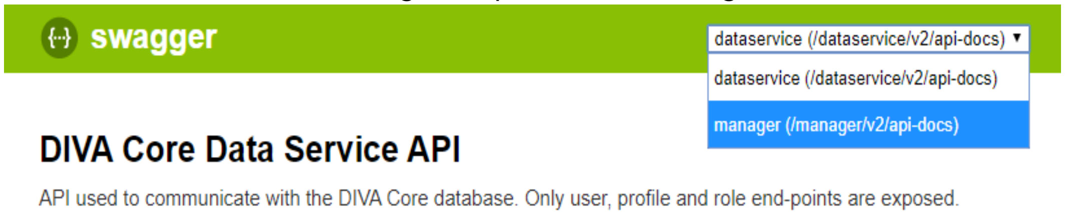
Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJleHAiOjE1ODcv		header	string
userReg	`{ "password": "password", "role": "admin", "username": "username" }`	userReg	body	Model | Example Value `{ "password": "string", "role": "string", "username": "string" }`

Parameter content type: application/json

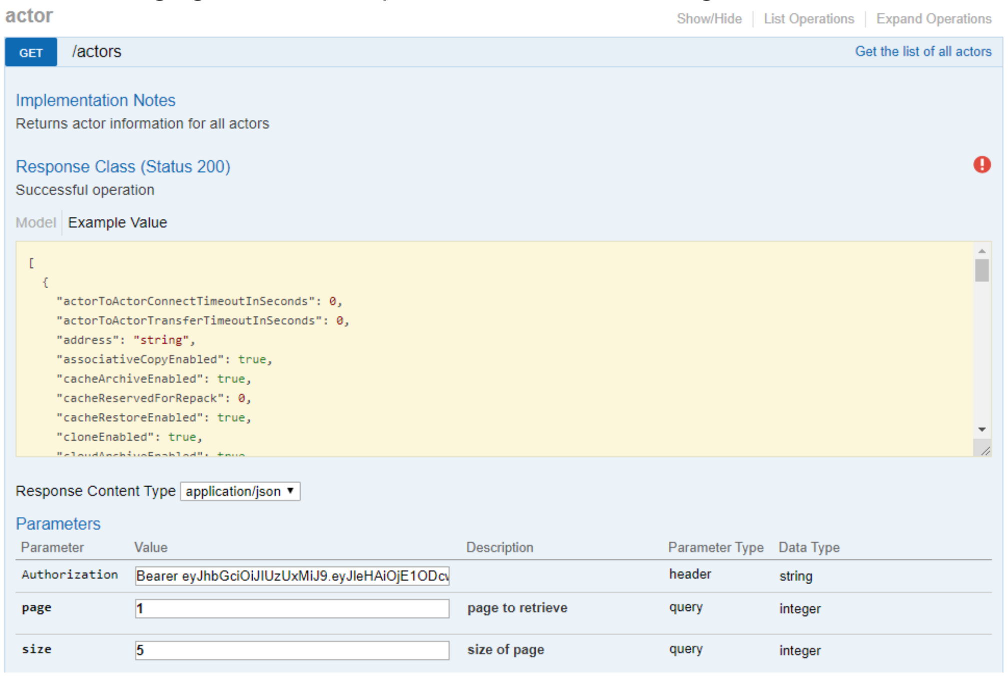
You are now ready to start using the API to retrieve information from DIVA Core. You need to switch to the Core Manager endpoints to start using the API.


swagger
dataservice (/dataservice/v2/api-docs)
dataservice (/dataservice/v2/api-docs)
manager (/manager/v2/api-docs)

DIVA Core Data Service API

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

The following figure is an example call to retrieve all configured Core Actors:



/actors — Get the list of all actors

Implementation Notes
Returns actor information for all actors

Response Class (Status 200)
Successful operation

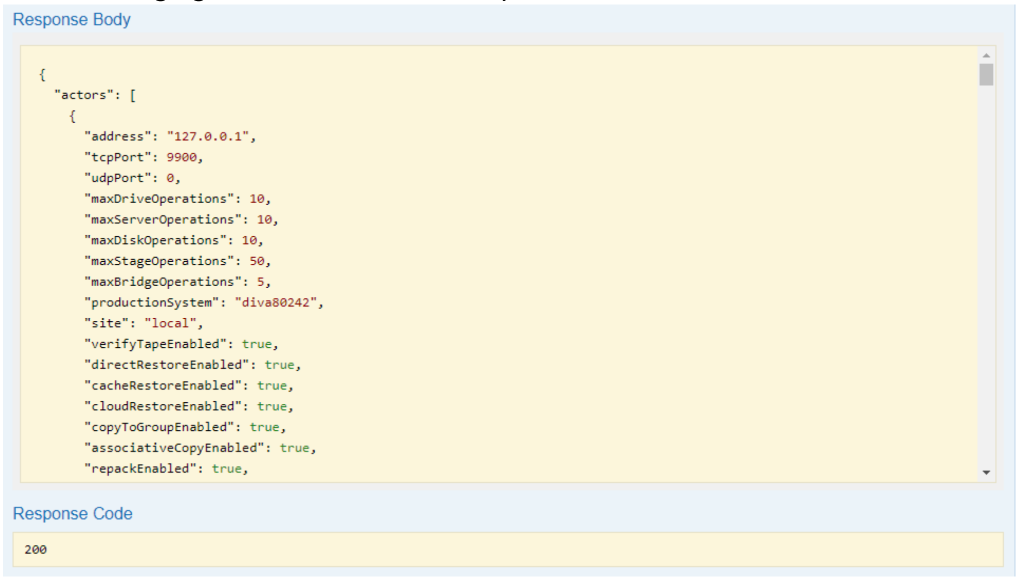
```
[
  {
    "actorToActorConnectTimeoutInSeconds": 0,
    "actorToActorTransferTimeoutInSeconds": 0,
    "address": "string",
    "associativeCopyEnabled": true,
    "cacheArchiveEnabled": true,
    "cacheReservedForRepack": 0,
    "cacheRestoreEnabled": true,
    "cloneEnabled": true,
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJleHAiOjE1ODcw		header	string
page	1	page to retrieve	query	integer
size	5	size of page	query	integer

The following figure is the start of the response:



Response Body

```
{
  "actors": [
    {
      "address": "127.0.0.1",
      "tcpPort": 9900,
      "udpPort": 0,
      "maxDriveOperations": 10,
      "maxServerOperations": 10,
      "maxDiskOperations": 10,
      "maxStageOperations": 50,
      "maxBridgeOperations": 5,
      "productionSystem": "diva80242",
      "site": "local",
      "verifyTapeEnabled": true,
      "directRestoreEnabled": true,
      "cacheRestoreEnabled": true,
      "cloudRestoreEnabled": true,
      "copyToGroupEnabled": true,
      "associativeCopyEnabled": true,
      "repackEnabled": true,
```

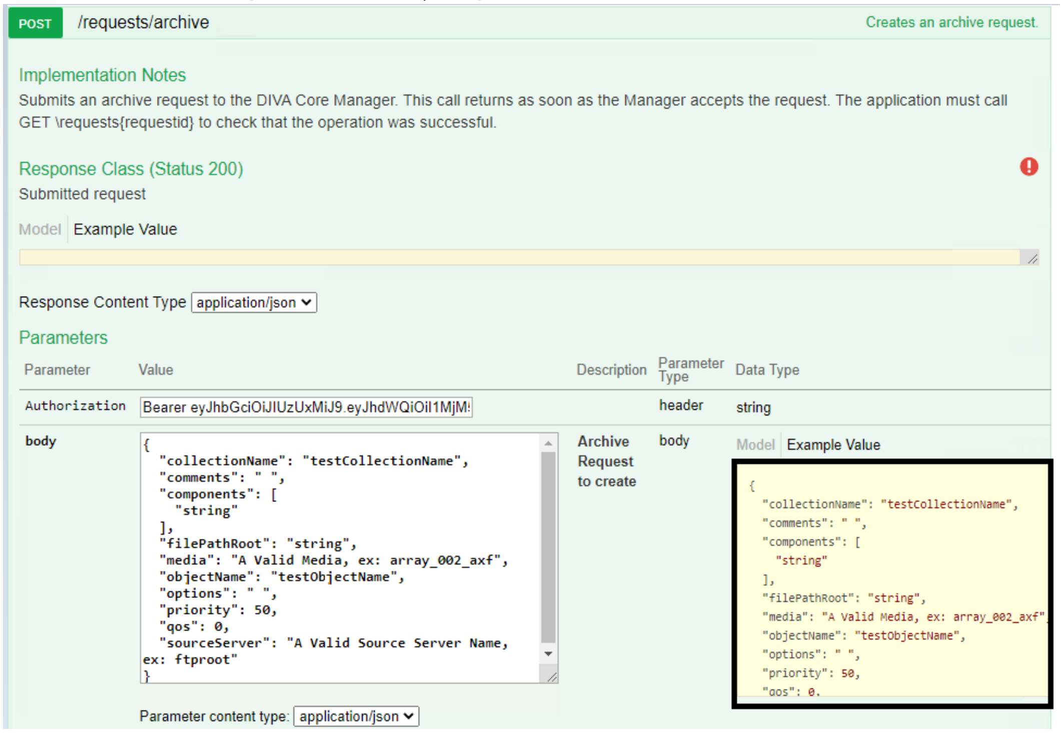
Response Code

```
200
```

To submit a request (for example an Archive request) you must submit a request to *POST /requests/archive*. The header must contain an Authorization Key with the bearer token as the value. The following is an example archive 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 that request and click on the Example Value to see all of the fields that must be specified for any request.



/requests/archive

Creates an archive request.

Implementation Notes

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.

Response Class (Status 200)

Submitted request

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJhdWQiOiI1MjM		header	string
body	(see below)	Archive Request to create	body	Model / Example Value

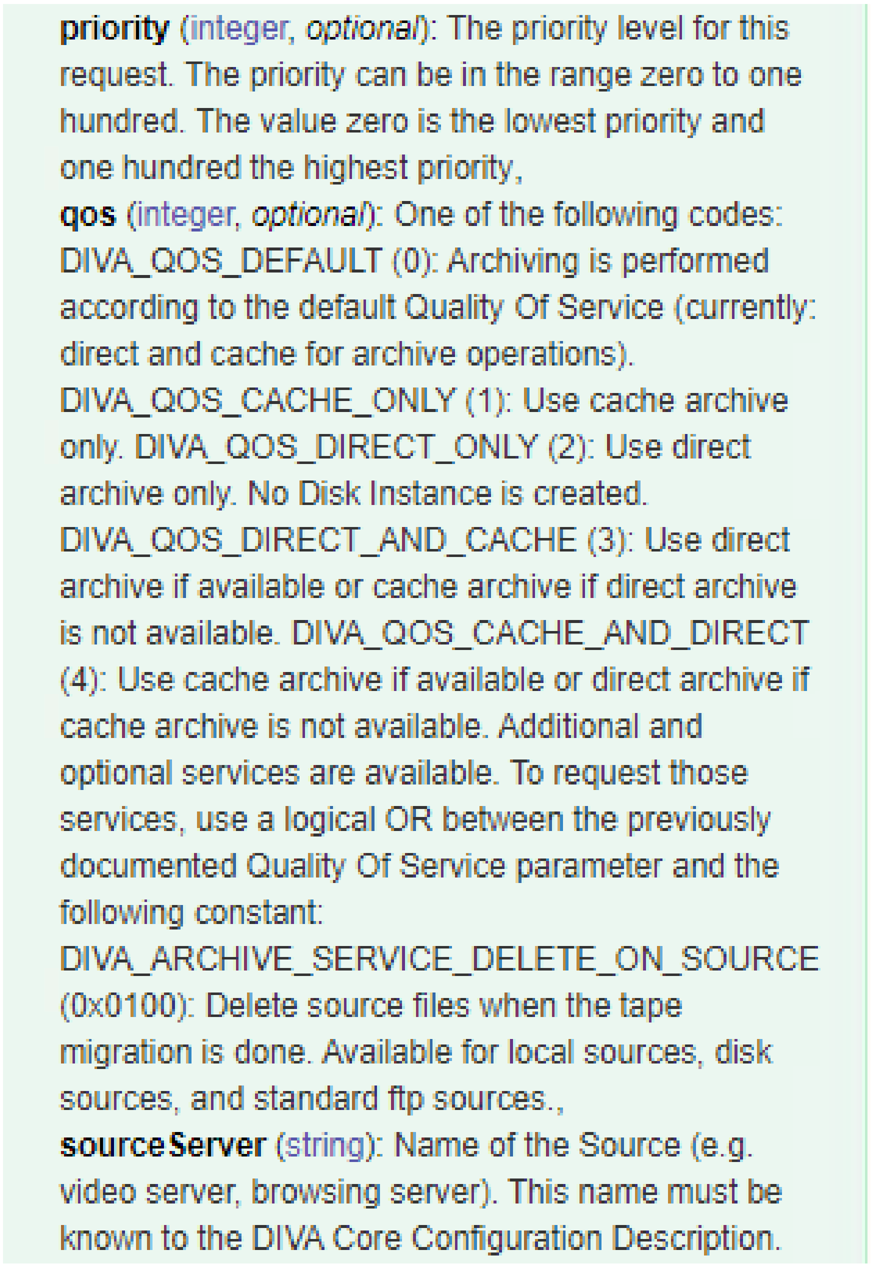
```
{
  "collectionName": "testCollectionName",
  "comments": " ",
  "components": [
    "string"
  ],
  "filePathRoot": "string",
  "media": "A Valid Media, ex: array_002_axf",
  "objectName": "testObjectName",
  "options": " ",
  "priority": 50,
  "qos": 0,
  "sourceServer": "A Valid Source Server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

```
{
  "collectionName": "testCollectionName",
  "comments": " ",
  "components": [
    "string"
  ],
  "filePathRoot": "string",
  "media": "A Valid Media, ex: array_002_axf",
  "objectName": "testObjectName",
  "options": " ",
  "priority": 50,
  "qos": 0,
```

Parameter content type: application/json

You can then specify the values and click Try it out.

Note: If you click on Model next to the Example Value tab it has a description of each field and a list possible values. For example, for qos, you'll see the list of possible QOS values and their meaning. A value of 2 signifies a QOS value of Direct-only.



priority (integer, *optional*): The priority level for this request. The priority can be in the range zero to one hundred. The value zero is the lowest priority and one hundred the highest priority,
qos (integer, *optional*): 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.,
sourceServer (string): Name of the Source (e.g. video server, browsing server). This name must be known to the DIVA Core Configuration Description.

Sample Program

The following is a sample program to get all Core Actors from DIVA Core in 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())
```