



DIVA

Application Programming Guide

Version 9.3.0

Copyrights and Trademark Notices

Copyright © 2024 Telestream, LLC and its Affiliates. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, altered, or translated into any languages without written permission of Telestream, LLC. Information and specifications in this document are subject to change without notice and do not represent a commitment on the part of Telestream. Specifications subject to change without notice.

Telestream, CaptionMaker, Cerify, DIVA, Episode, Flip4Mac, FlipFactory, Flip Player, GraphicsFactory, Kumulate, Lightspeed, MetaFlip, Post Producer, ScreenFlow, Switch, Tempo, TrafficManager, Vantage, VOD Producer, and Wirecast are registered trademarks and Aurora, ContentAgent, Cricket, e-Captioning, Inspector, iQ, iVMS, iVMS ASM, MacCaption, Pipeline, Sentry, Surveyor, Vantage Cloud Port, CaptureVU, FlexVU, PRISM, Sentry, Stay Genlock, Aurora, and Vidchecker are trademarks of Telestream, LLC and its Affiliates. All other trademarks are the property of their respective owners.

Adobe. Adobe® HTTP Dynamic Streaming Copyright © 2014 Adobe Systems. All rights reserved.

Apple. QuickTime, MacOS X, and Safari are trademarks of Apple, Inc. Bonjour, the Bonjour logo, and the Bonjour symbol are trademarks of Apple, Inc.

Avid. Portions of this product Copyright 2012 Avid Technology, Inc.

CoreOS. Developers of ETCD.

Dolby. Dolby and the double-D symbol are registered trademarks of Dolby Laboratories Licensing Corporation.

Fraunhofer IIS and Thomson Multimedia. MPEG Layer-3 audio coding technology licensed from Fraunhofer IIS and Thomson Multimedia.

Google. VP6 and VP8 Copyright Google Inc. 2014 All rights reserved.

MainConcept. MainConcept is a registered trademark of MainConcept LLC and MainConcept AG. Copyright 2004 MainConcept Multimedia Technologies.

Manzanita. Manzanita is a registered trademark of Manzanita Systems, Inc.

MCW. HEVC Decoding software licensed from MCW.

MedialInfo. Copyright © 2002-2013 MediaArea.net SARL. All rights reserved.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Microsoft. Microsoft, Windows Server 2016|Server 2019|Server 2022, Windows 10, Media Player, Media Encoder, .Net, Internet Explorer, SQL Server 2012|2016|2019|2022, and Windows Media Technologies are trademarks of Microsoft Corporation.

NLOG, MIT, Apache, Google. NLog open source code used in this product under MIT License and Apache License is copyright © 2014-2016 by Google, Inc., © 2016 by Stabzs, © 2015 by Hiro, Sjoerd Tieleman, © 2016 by Denis Pushkarev, © 2015 by Dash Industry Forum. All rights reserved.

SharpSSH2. SharpSSH2 Copyright (c) 2008, Ryan Faircloth. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer:

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Diversified Sales and Service, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Swagger. Licensed from SmartBear.

Telerik. RadControls for ASP.NET AJAX copyright Telerik All rights reserved.

VoiceAge. This product is manufactured by Telestream under license from VoiceAge Corporation.

x264 LLC. The product is manufactured by Telestream under license from x264 LLC.

Xceed. The Software is Copyright ©1994-2012 Xceed Software Inc., all rights reserved.

ZLIB. Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler.



Other brands, product names, and company names are trademarks of their respective holders, and are used for identification purpose only.

MPEG Disclaimers

MPEGLA MPEG2 Patent

ANY USE OF THIS PRODUCT IN ANY MANNER OTHER THAN PERSONAL USE THAT COMPLIES WITH THE MPEG-2 STANDARD FOR ENCODING VIDEO INFORMATION FOR PACKAGED MEDIA IS EXPRESSLY PROHIBITED WITHOUT A LICENSE UNDER APPLICABLE PATENTS IN THE MPEG-2 PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, LLC, 4600 S. Ulster Street, Suite 400, Denver, Colorado 80237 U.S.A.

MPEGLA MPEG4 VISUAL

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 VISUAL PATENT PORTFOLIO LICENSE FOR THE PERSONAL AND NON-COMMERCIAL USE OF A CONSUMER FOR (i) ENCODING VIDEO IN COMPLIANCE WITH THE MPEG-4 VISUAL STANDARD ("MPEG-4 VIDEO") AND/OR (ii) DECODING MPEG-4 VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL AND NON-COMMERCIAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION INCLUDING THAT RELATING TO PROMOTIONAL, INTERNAL AND COMMERCIAL USES AND LICENSING MAY BE OBTAINED FROM MPEG LA, LLC. SEE [HTTP://WWW.MPEGLA.COM](http://www.mpegla.com).

MPEGLA AVC

THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE [HTTP://WWW.MPEGLA.COM](http://www.mpegla.com).

MPEG4 SYSTEMS

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 SYSTEMS PATENT PORTFOLIO LICENSE FOR ENCODING IN COMPLIANCE WITH THE MPEG-4 SYSTEMS STANDARD, EXCEPT THAT AN ADDITIONAL LICENSE AND PAYMENT OF ROYALTIES ARE NECESSARY FOR ENCODING IN CONNECTION WITH (i) DATA STORED OR REPLICATED IN PHYSICAL MEDIA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND/OR (ii) DATA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND IS TRANSMITTED TO AN END USER FOR PERMANENT STORAGE AND/OR USE. SUCH ADDITIONAL LICENSE MAY BE OBTAINED FROM MPEG LA, LLC. SEE [HTTP://WWW.MPEGLA.COM](http://www.mpegla.com) FOR ADDITIONAL DETAILS.

Limited Warranty and Disclaimers

Telestream, LLC (the Company) warrants to the original registered end user that the product will perform as stated below for a period of one (1) year from the date of shipment from factory:

Hardware and Media—The Product hardware components, if any, including equipment supplied but not manufactured by the Company but NOT including any third party equipment that has been substituted by the Distributor for such equipment (the “Hardware”), will be free from defects in materials and workmanship under normal operating conditions and use.

Warranty Remedies

Your sole remedies under this limited warranty are as follows:

Hardware and Media—The Company will either repair or replace (at its option) any defective Hardware component or part, or Software Media, with new or like new Hardware components or Software Media. Components may not be necessarily the same, but will be of equivalent operation and quality.

Software Updates

Except as may be provided in a separate agreement between Telestream and You, if any, Telestream is under no obligation to maintain or support the Software and Telestream has no obligation to furnish you with any further assistance, technical support, documentation, software, update, upgrades, or information of any nature or kind.

Restrictions and Conditions of Limited Warranty

This Limited Warranty will be void and of no force and effect if (i) Product Hardware or Software Media, or any part thereof, is damaged due to abuse, misuse, alteration, neglect, or shipping, or as a result of service or modification by a party other than the Company, or (ii) Software is modified without the written consent of the Company.

Limitations of Warranties

THE EXPRESS WARRANTIES SET FORTH IN THIS AGREEMENT ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. No oral or written information or advice given by the Company, its distributors, dealers or agents, shall increase the scope of this Limited Warranty or create any new warranties.

Geographical Limitation of Warranty—This limited warranty is valid only within the country in which the Product is purchased/licensed.

Limitations on Remedies—YOUR EXCLUSIVE REMEDIES, AND THE ENTIRE LIABILITY OF TELESTREAM, LLC WITH RESPECT TO THE PRODUCT, SHALL BE AS STATED IN THIS LIMITED WARRANTY. Your sole and exclusive remedy for any and all breaches of any Limited Warranty by the Company shall be the recovery of reasonable damages which, in the aggregate, shall not exceed the total amount of the combined license fee and purchase price paid by you for the Product.

Damages

TELESTREAM, LLC SHALL NOT BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OR INABILITY TO USE THE PRODUCT, OR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, EVEN IF THE COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF THOSE DAMAGES, OR ANY REMEDY PROVIDED FAILS OF ITS ESSENTIAL PURPOSE.

Further information regarding this limited warranty may be obtained by writing:
 Telestream, LLC
 848 Gold Flat Road
 Nevada City, CA 95959 USA

You can call Telestream during U. S. business hours via telephone at (530) 470-1300.

Regulatory Compliance

Electromagnetic Emissions: FCC Class A, EN 55022 Class A, EN 61000-3-2/-3-3, CISPR 22 Class A

Electromagnetic Immunity: EN 55024/CISPR 24, (EN 61000-4-2, EN 61000-4-3, EN 61000-4-4, EN 61000-4-5, EN 61000-4-6, EN 61000-4-8, EN 61000-4-11)

Safety: CSA/EN/IEC/UL 60950-1 Compliant, UL or CSA Listed (USA and Canada), CE Marking (Europe)

California Best Management Practices Regulations for Perchlorate Materials:
 This Perchlorate warning applies only to products containing CR (Manganese Dioxide) Lithium coin cells. Perchlorate Material-special handling may apply. See www.dtsc.ca.gov/hazardouswaste/perchlorate.

Obtaining Support | Information | Assistance

Contact Telestream for support, information or assistance, as indicated.

Resource	Contact Information
DIVA Technical Support Information Assistance FAQ Forums Upgrades	<p>Web site: telestream.net/telestream-support/DIVA/support.htm</p> <p>Web site: telestream.net/telestream-support/DIVA/support.htm</p> <p>Support Email: support@telestream.net</p> <p>US Phone: 877-257-6245</p> <p>International Phone: +1 530-470-2036</p> <p>Support hours: Monday - Friday, 7am - 6pm Pacific time.</p> <p>P1 Support: 24 x 7.</p> <p>Terms and times of support services vary, per the terms of your current service contract with Telestream.</p> <p>Depending on problem severity, Telestream responds within 24 business hours. For P1 issues, Telestream responds within 1 hour. Please see the Maintenance & Support Guide for definitions.</p>
Product Licensing	<p>Web site: telestream.net/telestream-support/DIVA/support.htm</p> <p>License Email: license@telestream.net</p>
Telestream, generally	<p>Web site: telestream.net</p> <p>Sales and Marketing Email: info@telestream.net</p>
Telestream Reseller Support	If you purchased your product from a reseller, please contact your reseller for support.
International Reseller Support	<p>Website: telestream.net</p> <p>See the website for your regional authorized Telestream reseller.</p>
Telestream Technical Writers	<p>Email: techwriter@telestream.net</p> <p>If you have comments or suggestions about improving this document or other Telestream documents—or if you've discovered an error or omission, please email us.</p>

DIVA Documentation Library

For more information about DIVA, browse the DIVA Documentation Library at telestream.net/telestream-support/DIVA/support.htm.

Contents

Introduction 11

DIVA Installation Instructions	11
DIVA Jobs (Requests)	11
Archive Job Terms	12
Restore Job Terms	13
Partial Restore Job Terms	13
Delete Object Job Terms	13
Operational Boundaries	14
Number of DIVA Connections	14
Number of Simultaneous DIVA Jobs	14
Number of API Tasks	14
Special Authorized Characters	14
Maximum Number of Allowed Characters	16
File Path Limitations	16

The DIVA REST API 17

Main DIVA REST API Calls	17
Start Configuring the DIVA REST API	18
Sample Python Program Using DIVA REST API	20
DIVA API Secure Communications	21
Secure Sockets Layer (SSL) and Authentication	21
DIVA Data Service API	23
DIVA Connect REST API	24
DIVA REST API Workflows	24
Authentication Token Workflow	25
Roles	25
DIVA REST API Workflow Guidelines	26
DIVA Job Status Codes	27
Partial Restore Job Formats and Manager Responses	29

Python DIVAScript 33

Python DIVAScript	33
-------------------	----

DIVAScript Commands Supported by DIVA REST API	33
DIVAScript Configuration File Settings	34
Running DIVAScript in Server Mode	34
DIVAScript Command-Line Mode Requirements	34
DivaScript Command Syntax	35

The DIVA Java API 39

Viewing Java API Documentation Before Installation	39
Location of DIVA Java API Documentation	40
DIVA Java API Known Issues	40

The DIVA C++ API 41

The DIVA C++ API	41
DIVA C++ API Requirements	42
DIVA Release Compatibility	44
Managing Connections	44
Using the DIVA C++ API in Multithreaded Applications	44
DIVA C++ API Unicode Strings	45
Session Management Commands	45
DIVA_getApiVersion	45
DIVA_SSL_initialize	46
DIVA_connect	46
DIVA_disconnect	48
Jobs and Commands	49
DIVA_addGroup	49
DIVA_archiveObject	50
DIVA_associativeCopy	54
DIVA_cancelRequest	56
DIVA_changeRequestPriority	57
DIVA_copyToGroup and DIVA_copy	59
DIVA_copyToNewObject	62
DIVA_deleteGroup	66
DIVA_deleteInstance	67
DIVA_deleteObject	70
DIVA_ejectTape	72
DIVA_enable_Automatic_Repack	73
DIVA_getArchiveSystemInfo	74
DIVA_getArrayList	79
DIVA_getFinishedRequestList	82
DIVA_getFilesAndFolders	84
DIVA_getGroupsList	88
DIVA_getObjectDetailsList	89
DIVA_getObjectInfo	103
DIVA_getPartialRestoreRequestInfo	104
DIVA_getRequestInfo	105
DIVA_getSourceDestinationList	111

DIVA_getStoragePlanList	113
DIVA_getTapeInfo	114
DIVA_insertTape	115
DIVA_linkObjects	117
DIVA_lockObject	118
DIVA_multipleRestoreObject	119
DIVA_partialRestoreObject	123
DIVA_release	135
DIVA_require	136
DIVA_restoreInstance	138
DIVA_restoreObject	141
DIVA_transcodeArchive	145
DIVA_transferFiles	148
DIVA_unlockObject	150

Introduction

This guide provides information and instruction for software developers who create applications using DIVA. This guide contains information about configuring and operating DIVA for various programming architectures. Consult it when installing, configuring, or operating DIVA for any of these architectures.

Topics

- [DIVA Installation Instructions](#)
- [DIVA Jobs \(Requests\)](#)
- [Operational Boundaries](#)

DIVA Installation Instructions

For instructions on installing DIVA, see the *DIVA Installation and Administration Guide* on the Telestream website, here: [Need Help with DIVA?](#)

DIVA Jobs (Requests)

Archive, Restore and Delete are the basic DIVA jobs. The core concept of DIVA is data preservation.

Note: Telestream strongly recommends using the DIVA REST API rather than older APIs such as DIVA Enterprise Connect API, DIVAS API, DIVA Java API and DIVA C++ API. Although DIVA still supports the older APIs, the REST API, which is integrated into DIVA, offers enhanced features. Also, the DIVA Web App requires the DIVA REST API to function.

Note: In DIVA programming, we now use the term *job* instead of *request*. *Job* is also the official term used in programming for the DIVA Web App and all DIVA documentation.

However, documentation for earlier versions of DIVA, which use *request*, still uses the term *request*.

Archive Job Terms

Object

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

Identified an object by a Name and a Collection (category). Choose names for Object Name and Collection as desired. DIVA checks only that the Object Name + Collection combination is unique.

Collection

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

Files

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

Media Name is the DIVA device used for storing the object. The device 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 retrieve the list of all Arrays and Tape Groups can from DIVA, without necessarily exposing these to the end user. Check with the customer and the DIVA Project Manager about which Media to expose to the end user.

You can also use a Storage Plan as a storage device. For details, see the *DIVA Storage Policy Manager* guide.

Unmanaged Storage Repository Name

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.

Source Path Root

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

Note: You can obtain the source list using the `GET /servers` API call.

Quality of Service

You can leave the Quality of Service parameter at the default setting.

Priority

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

Delete From Source

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 Job Terms

You must specify the following terms for a Restore Job:

- 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 job and will overwrite the object if it already exists, unless the Do Not Overwrite option is specified.

Partial Restore Job Terms

The Partial Restore terms are the same as the Restore terms 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 Object Job Terms

A Delete Object Job deletes 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.

Operational Boundaries

Number of DIVA Connections

The number of connections to DIVA is limited by DIVA and set in the DIVA configuration file. The default configuration limit is 200. This includes connections to GUIs, Actors and all API clients. When the configured limit is reached, the API doesn't create additional connections.

See the `diva.conf` and `manager.conf` configuration files for more information.

Number of Simultaneous DIVA Jobs

The maximum number of simultaneous jobs processed by DIVA is configurable in the `diva.conf` file as the value of the `DIVA_MAX_SIMULTANEOUS_REQUESTS` parameter. The default value has been raised from two hundred to five hundred. The maximum number has been verified up to two thousand. Additional simultaneous jobs beyond the value set in this parameter are rejected by DIVA.

Number of API Tasks

The number of API tasks that will be accepted to the API Processing Queue is configurable in the `diva.conf` file as the value of the `DIVA_API_TASK_QUEUE_SIZE` parameter. The default value is two thousand and DIVA has been verified at this value. If the queue is full, subsequent commands are rejected.

Special Authorized Characters

Many jobs require alpha-numeric text terms. Special characters can be used in these fields as defined in the following table. The job is rejected if an invalid special character is used. In a Windows environment, file and folder names can't consist of one or more spaces, and can't contain a double-quotation mark.

Field (across) Character (down)	Name	Collection	Source	Media	Path	File	Comments	Options
~	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
'	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
!	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
@	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
#	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
\$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Field (across) Character (down)	Name	Collection	Source	Media	Path	File	Comments	Options
%	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
^	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
&	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
*	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
(Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
_	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
+	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
=	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
\	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
}	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
{	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
[Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
:	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
;	Yes	Yes	Yes	Yes	Yes ¹	Yes	Yes	Yes
"	Yes	Yes	Yes	Yes	No	Yes	Yes	No
'	Yes	Yes	No	No	Yes ¹	Yes	Yes	Yes
<	Yes	Yes	Yes	Yes	No	Yes	Yes	No
,	Yes	Yes	Yes	Yes	Yes ¹	Yes	Yes	Yes
>	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
.	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
?	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
/	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Space	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes

1. Depends on file system restrictions.

Maximum Number of Allowed Characters

There is a maximum number of characters you can use for each job term. If these limits are exceeded, the job is rejected. The maximum number of characters per term is listed below:

Job Term	Maximum # of Characters
Name	192
Collection	96
Source	96
Media	96
Path and File Name	1536
Comments	4000
Options	768

File Path Limitations

DIVA supports absolute path names on both Windows up to a maximum of 4000 characters. Relative path names are limited to 32676 characters on Windows systems (only).

A DIVA Windows local path is structured in the following order and terminated with a NUL character:

`Drive_Letter:\Component_Name\Component_Name\File_Name.Extension`

Here are example paths used in DIVA in Windows. The <NUL> character used in the example represents the invisible terminating null character for the current system code page and need not be entered. (The < and > characters are used to identify the NUL character by name, and must not be part of a path string).

Generic Path:

`D:\Some_32676-Character_Path_String<NUL>`

Actor Executable:

`D:\diva\diva\Program\Actor\bin\diva.exe`

DIVA Configuration File:

`D:\diva\10\Program\conf\diva\diva.conf`

The DIVA REST API

This chapter provides details about programming the DIVA REST API.

Topics

- [Main DIVA REST API Calls](#)
- [DIVA Data Service API](#)
- [DIVA REST API Workflows](#)

Main DIVA REST API Calls

The following are the main DIVA REST API calls available. These are the minimum required to implement the basic DIVA REST 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>

or from the Telestream DIVA Support Portal:

<https://www.telestream.net/telestream-support/diva/support.htm>

DIVA REST APIs on Swagger

The REST API Swagger page is home to all DIVA REST APIs. You can find other APIs using the Definition drop-down menu on the Swagger page.

The Swagger documentation for the REST API services is accessible using the following URL by replacing `localhost` with the correct IP address:

<https://localhost:8765/webjars/swagger-ui/index.html>

Also, the DIVA REST API documentation is included in DIVA as HTTP documentation. You can find the REST API documentation directly within the REST API.

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

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

Start Configuring the DIVA REST API

To start configuring the DIVA REST API, do the following:

1. Obtain all automations and DIVA REST API calls that use these credentials from the person who created the user.

During installation, a user is created by either the DIVA Installer, or manually by an administrator.

2. Call the `POST users/login` endpoint.

DIVA opens the `POST users/login` dialog.

3. In the `userLogin` field, specify the login and password.

For example:

```
"emailAddress": "a@abc.com",
"password": "password",
```

4. Click the *Execute* button.

DIVA sends you a token.

5. In the *Responses* window, copy the contents of the bearer token (everything in quotes after `"Token":`)

6. To create a user, enter the token in the *Authorization* field.

7. In the *userReg* field, enter the user name, password, and role of the user.

For example:

```
"emailAddress": "a@abc.com",
"password": "password",
"role": "operator",
```

8. (Optional) Set a timeout value between 0 and 10080 minutes. Default: 1440 minutes.

For example:

```
"sessionTimeout": 99999999
```

The session timeout is the number of minutes the user remains logged into DIVA.

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

- All DIVA `GET` jobs require at least the `user` role.
- Archive, Restore (including N-Restore and Partial Restore) and Copy jobs require at least the `operator` role.
- Change Priority, Transfer, Eject, Insert, Export and Import jobs require at least the `advoperator` role.
- All other jobs require the `administrator` role.

You can now use the DIVA REST API to retrieve information from DIVA. Switch to the Manager endpoints to start using the DIVA REST API.

Retrieving All Configured Actors

To retrieve all configured Actors, do the following:

1. To get a list of all actors in the configuration, call `GET/actors`.
DIVA opens the `GET/actors` dialog.
2. Click the *Execute* button.
DIVA displays a list of actor settings in a *Responses* dialog.

Submitting a Job

To submit a job (for example, an Archive job, do the following:

1.) Submit `POST /requests/archive`.
DIVA opens the `POST /requests/archive` dialog.
2. (Required) Enter an authorization key into the `Authorization` string field with the bearer token as the value, as depicted in this job:

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' --header 'Authorization: Bearer
eyJhbGciOiJIUzUx
MiJ9.eyJhdWQiOiI1MjM5YTcxOSliYjAwLTQ5MWQtOGYxZi01ZjcxM2YxZWZiMj
MiLCJleHAiOiJlMjM5ZjZNTY0MDcsImhhdCI6MTYyMTI3MDAwNywiYXV0aG9yaXRpZ
```

```

XMiOlsic3lzYWRtaW4iXSwidXNlcm5hbWUiOiJzeXNhZG1pbiJ9.zZiKvEe-
3JjuOsJ-CDpW_32JKRefy54-wGwra_LABmUeuIhpWGEpHnT-Se5PXTFxv-
jDf2g9mgezKQIvIJzObzQ' -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://xxx.xx.xx.xx:xxxx/manager/requests/archive'

```

3. Go to the Swagger page for the job and click on the Example Value.

Swagger opens the `POST/requests/archive` dialog, displaying the fields that must be specified for any job.

4. Specify the values for the *Authorization string* and *body* fields.

5. To submit the job, click *Execute*.

DIVA opens the `POST/requests/archive` dialog.

6. To view a description of each field and a list possible values, click *Model* (next to the *Example Value* tab).

DIVA opens the *Model* dialog.

For example, DIVA displays the list of possible QOS values and their meaning. A value of 2 signifies a QOS value of Direct-only.

Sample Python Program Using DIVA REST API

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())

```

DIVA API Secure Communications

The following sections describe secure communications for one of the DIVA APIs. The DIVA Java and DIVA C++ Initiators use the default keys and certificates file in the %DIVA_API_HOME%/Program/security folder when connecting to the Manager.

The Manager Service is backward compatible with earlier versions of the DIVA Java API, the DIVA C++ API, the DIVA Web Services API, Enterprise Connect 1.0, and DIVA Connect. The Manager Service establishes connections over regular sockets. The earlier DIVA Java API and DIVA C++ API releases can establish Manager communications using either secure or insecure sockets. Only the Manager supports secure communications.

The Manager Service and DIVA Connect 4.0 support both secure and insecure communication ports simultaneously. The Manager default secure port is tcp/8000, and the default insecure port is tcp/9000.

The DIVA C++ API includes a new call named `DIVA_SSL_initialize` added to set the environment for secure communication with the Manager Service.

`DIVA_SSL_initialize` must be called before calling DIVA Connect with DIVA. Otherwise the DIVA Connect call fails.

Secure Sockets Layer (SSL) and Authentication

DIVA consists of services in both Java and C++. Java and C++ represent certificates and keys differently from one another. DIVA has the keys and certificates for Java services in a Java Keystore file. DIVA has the keys and certificates for C++ services in Privacy Enhanced Mail (PEM) format files.

When connecting to the DIVA WebUI for the first time, there is usually a security error page displayed by the web browser. This error means that the HTTPS server certificate is not trusted by the browser. This is the certificate for the DIVA REST API Gateway (DIVA\Program\security\certificates\RestAPIService.p12). This issue is caused by the fact that the certificates generated by DIVA were self-signed. This is verifiable by showing the certificate because it has been issued "by" and "to" the same organization.

You may accept the risk and continue to connect, but you will always get the same error for every new connection.

The DIVA security tool (DIVA\Program\security\bin\DIVASecurityTool.bat) has been fixed to generate certificates signed DIVA certificate authority (DIVA_CA). With the new security tool, the new certificates are no longer self-signed.

Before applying the security tool (before DIVA 9.0), make sure to make a backup copy of DIVA\Program\DIVA_CA\DIVA_CA.cnf because it contains the list of domains or IP addresses to connect to the Web App. If the Web App is being accessed using https://IP_Address/DIVASecurityTool/login, the IP address must be listed in the alt_names section. This also applies to domain names or hostnames. The second security tool option will automatically add the IP address and the hostname of the server to the alt_names section is at the end of the file.

With the fixed security tool, you must generate new certificates (option 2) and restart all the DIVA services. Contact Telestream Technical Support for assistance as necessary.

All internal DIVA services (Web App, DBBackup, Migration Utility, Actor, SPM, WFM, SNMP, Robot Manager, RDTU, and Migration Services) can only connect to secure ports. The Web App will report an SSL Handshake Timeout if you attempt to connect to the non-secure port. Clients using the DIVA Java API or DIVA C++ API are allowed to connect to either port.

The following is a relative snippet from the Manager configuration file:

```
# Port number on which the DIVA is waiting for incoming
connections.
# Note: If you are using a Sony Managed Storage and plan to execute
the DIVA
# on the same machine as the PetaSite Controller (PSC) software, be
aware
# that the PSC server uses the 9000 port and that this cannot be
modified.
# In that situation, you have to use a different port for the DIVA.
# This same warning applies to FlipFactory which uses ports 9000
and 9001.
# The default value is 9000.
DIVAMANAGER_PORT=9000


# Secure port number on which the DIVA is waiting for incoming
connections.
# The default value is 8000.
DIVAMANAGER_SECURE_PORT=8000A new folder called %DIVA_API_HOME%/
security is added to the API installation structure as follows:
%DIVA_API_HOME%
    security
        conf
```

The conf folder contains the SSLSettings.conf file that is used to configure the SSL handshake timeout.

DIVA Data Service API

The DATA Service API communicates with the database. Only user, profile, and endpoints are exposed.

Use the DIVA Core Data Service Swagger page 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.


Swagger
Supported by SMARTBEAR

Select a definition

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.

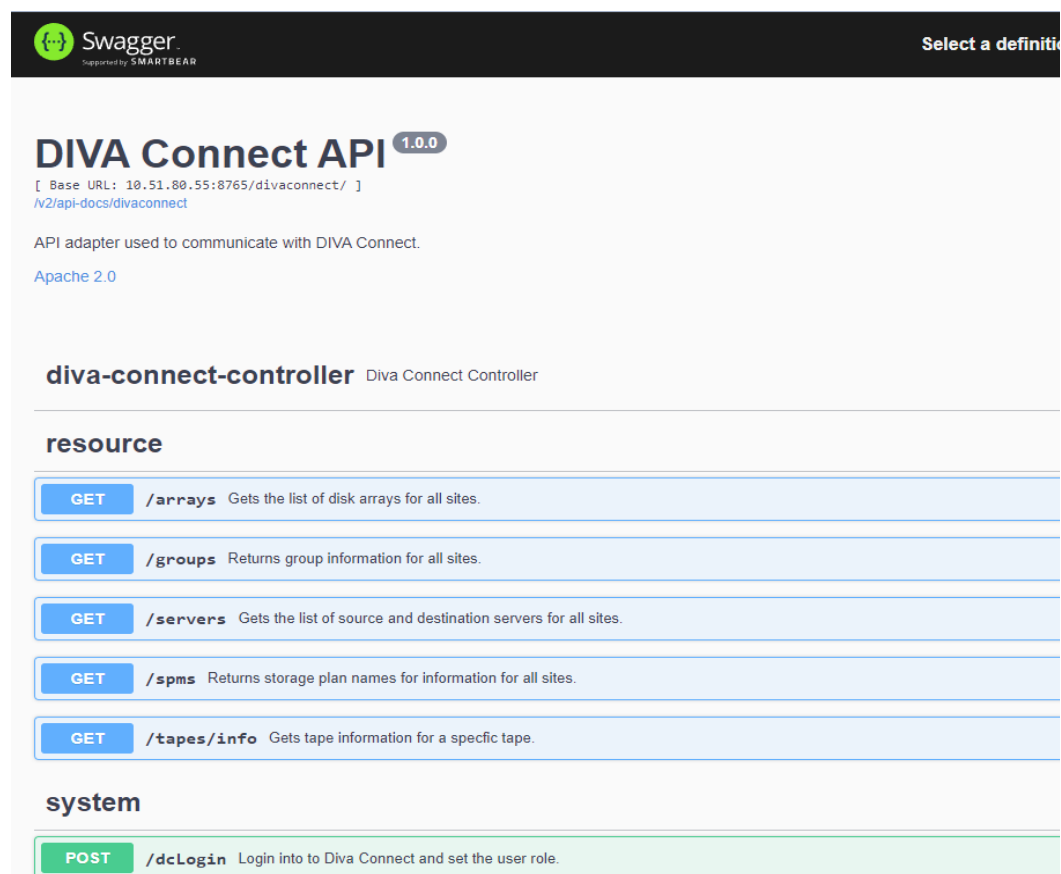
DIVA Connect REST API

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

During installation, a user is created either by the DIVA Installer, or manually by an administrator. Obtain all automations and API calls that use these credentials, from the person who created the user. Go to the POST users/login.

For detailed information, see the DIVA Connect documentation on the DIVA Support Portal at <https://www.telestream.net/telestream-support/diva/support.htm>

Swagger contains DIVA Connect REST API endpoints as shown here:



The image shows the Swagger UI for the DIVA Connect API. The header includes the Swagger logo and the text "Supported by SMARTBEAR". The main title is "DIVA Connect API" with a version badge "1.0.0". Below the title, it shows the base URL: "[Base URL: 10.51.80.55:8765/divaconnect/]" and a link to the API docs: "/v2/api-docs/divaconnect". It also mentions the API adapter used: "API adapter used to communicate with DIVA Connect. Apache 2.0".

The main section is titled "diva-connect-controller" with the subtitle "Diva Connect Controller". Under the "resource" section, there are five GET endpoints:

- 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 specific tape.

Under the "system" section, there is one POST endpoint:

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

DIVA REST API Workflows

This chapter describes the DIVA REST API and Authentication Token Workflows. The DIVA REST API uses JWT (JSON Web Token) authentication specified in the authorization header of all jobs. 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 DIVA 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 first call to a DIVA REST API, and then use that token as long as it is valid. Any HTTP job 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:

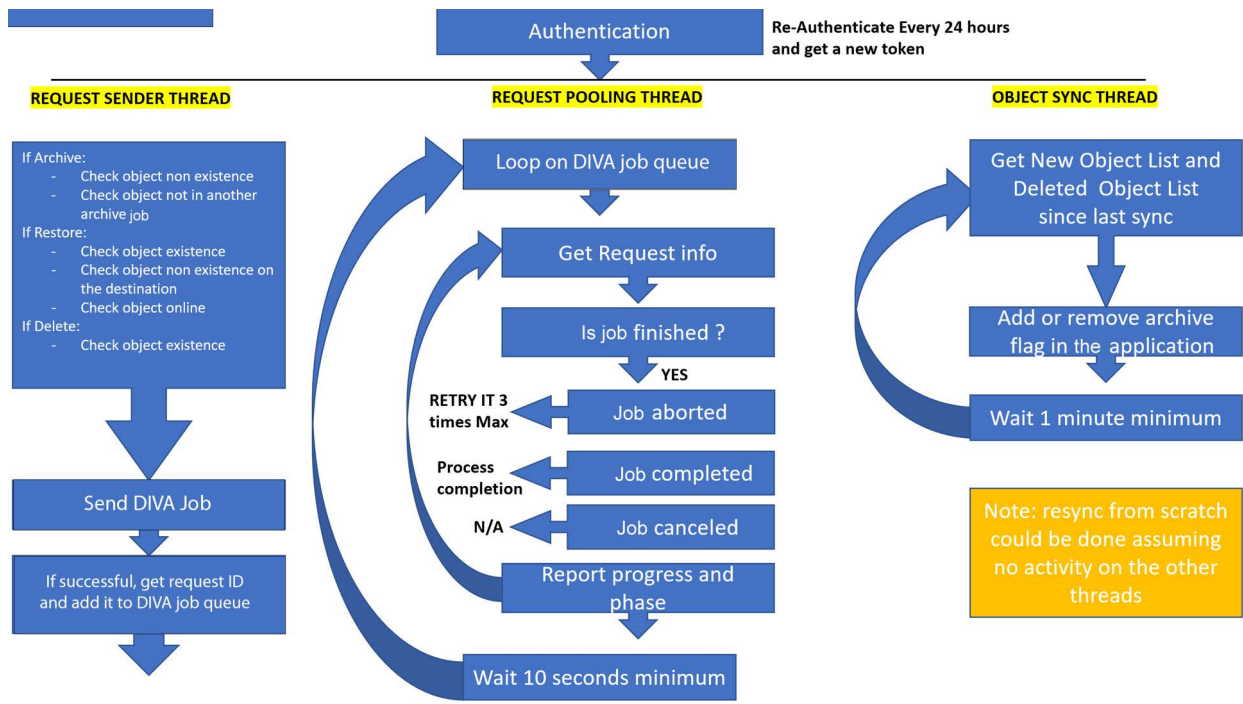
- POST /users
- DELETE /users
- GET /users
- GET /roles

DIVA REST API Workflow Guidelines

Use the following guidelines to develop workflows using the DIVA REST 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 job.
- Send the DIVA job (Archive, Restore, and so on) using the token from the last step and get the Request ID. Add the Request ID to the DIVA job queue.
- Pool every n seconds on the DIVA job queue list using `getRequestInfo`. Wait a minimum of 10 seconds between each pooling phase.
- The progress and phase can be obtained for each running job.
- Any running job can be canceled.
- You can remove a finished job from the DIVA job queue. A finished job will be COMPLETED, PARTIALLY_COMPLETED, ABORTED, or CANCELLED.
- Avoid retrying too many times if a job 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:



DIVA Job Status Codes

This table describes DIVA job 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

Code	Name	Description
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
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

Code	Name	Description
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 Job Formats and Manager Responses

The following formats—each identified by an INT value in *format*—are used when issuing jobs 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)

Job and Response Sample

Here are Partial Restore jobs 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": [
    {
      "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"
}

```


Python DIVAScript

This chapter describes DIVAScript for Python via the DIVA REST API.

Topics

- [Python DIVAScript](#)
- [Running DIVAScript in Server Mode](#)
- [DIVAScript Command-Line Mode Requirements](#)

Python DIVAScript

DivaScript was developed in python. DIVAScript supports sending commands via the DIVA REST API. DIVAScript helps develop and execute automated functional tests via the DIVA REST API.

DivaScript supports most DIVArchive commands that are available via the DIVA REST API.

DIVAScript Commands Supported by DIVA REST API

- login
- logout
- addgroup
- objectinfo
- objlist
- reqinfo
- version
- archive
- restore
- partialrestore

- copy
- copyas
- delete
- trarchive
- scan
- stage

DIVAScript Configuration File Settings

```
diva_serv: serve_ip
diva_port: 8765
diva_user: diva_user
diva_pass: diva_pass
protocol: https
diva_version: 9
```

Running DIVAScript in Server Mode

To run DIVAScript in server mode, do the following:

1. Open the Windows command-line interface.
2. Type DIVAScript.exe and press Enter.

DIVAScript displays the list of valid DIVAScript terms. DIVAScript doesn't require a parameter.

DIVAScript Command-Line Mode Requirements

The requirements for running DIVAScript in Command-Line Mode follow:

You must first call the login with correct credentials so that the token is saved and used for next commands.

If a parameter value contains spaces or other special characters, place the value inside " "

Use the pipe (|) delimiter for terms that allow a list of values (e.g: filelist from archive or in / out from partialRestore).

Command line syntax is DivaScript.exe following commandName: one of the recognized command names, like `archive` or `delete`.

Other terms: depending on the command, additional terms may be required.

The command name must be the first parameter. All other terms can be placed anywhere in the command line.

DivaScript Command Syntax

login

Login a user, returns a created token

Synopsis

DivaScriptPy.exe login -u USERNAME -p PASSWORD

logout

Logout a user, returns the status.

Synopsis

DivaScriptPy.exe logout -u USERNAME

addgroup

Create a tape group in the configuration.

Synopsis

DivaScriptPy.exe addgroup [-aqos ARCHIVEQOS] [-cpr COMPRESSIONENABLED] [-cmt DESCRIPTION] [-en ENCRYPTIONENABLED] [-fmt FORMAT] [-grp NAME] [-nrt NUMBEROFTAPESUSEDFORREPACK] [-prio PRIORITY] [-rqos RESTOREQOS] [-set SET] [-repk USEDFORREPACK] [-vw VERIFYWRITE] [-wf WORSTFITENABLED]

objinfo

Gets object information for a specific object.

Synopsis

DivaScriptPy.exe objinfo -obj OBJECT -col COLLECTION

objectsList

Gets object information for all objects based on filter criteria. Returns a list of objects. On success, print one line for each matching object (name@category).

Synopsis

DivaScriptPy.exe objlist [-time INITIALTIME] [-lt LISTTYPE] [-objlt OBJECTSLISTTYPE] [-maxs MAXLISTSIZE] [-obj OBJECTNAME] [-col COLLECTIONNAME] [-grp MEDI-ANAME] [-lod LEVELOFDETAIL]

reqinfo

Get request by ID. Return the current status of a DIVA request.

On success, print the current status of the request (first line is the numerical status; second line is the text status in capital letters).

If this command is issued with the “wait” parameter, the getRequestInfo command will continue to run until the request related to the entered Request ID completes.

This parameter is important when it is necessary to wait for one DIVA request to complete before another is submitted via DivaScript.

Synopsis

DivaScriptPy.exe reqinfo -req REQUESTID [-wait WAIT]

version

Gets DIVA build version and API version. Returns the DIVA build version and the API version.

Synopsis

DivaScriptPy.exe version

archive

Submits an archive request to the DIVA Core Manager. This call returns as soon as the Manager accepts the request.

Use reqinfo command to know when the request has been executed.

On success, divascript prints the request ID on the output.

Synopsis

Divascript.exe archive -obj OBJECT -col COLLECTION -grp MEDIA_GROUP -src SERVER -fpr FILES_PATH_ROOT -filelist FILELIST [-com COMMENTS]

The following arguments are required: -obj/--object, -col/--collection, -grp/--media_group, -src/--server, -fpr/--files_path_root, -filelist/--filelist

When there multiple files, use pipe delimiter: -filelist "test.mov|gxf-sample.gxf"

DivaScriptPy.exe archive -col COLLECTIONNAME [-com COMMENTS] -filelist COMPONENTS -fpr FILEPATHROOT -grp MEDIA -obj OBJECTNAME [-opt OPTIONS] [-prio PRIORITY] [-qos QOS] -src SOURCESERVER archive

restore

Submits an Object Restore request to the Manager and the Manager chooses the appropriate instance to be restored. If the requested object is on media that is not available, the request will fail.

Use reqinfo command to know when the request has been executed.

On success, divascript prints the request ID on the output

Synopsis

DivaScriptPy.exe restore -col COLLECTIONNAME -src DESTINATIONSERVER -fpr FILEPATHROOT [-inst INSTANCE] -obj OBJECTNAME [-opt OPTIONS] [-prio PRIORITY] [-qos QOS]

partialrestore

Submits a Partial Object Restore request to the Manager and the Manager chooses the appropriate instance to be restored. If the request was not accepted (for example, if the requested object is on media not currently available) the request will generate an error. The Manager will use the instanceID field to select the instance of the object to use for

the Partial Restore operation. The Manager will choose an appropriate instance to restore if -1 is used.

Use reqinfo command to know when the request has been executed.

On success, divascript prints the request ID on the output

Synopsis

```
DivaScriptPy.exe partialrestore -obj OBJECTNAME -col COLLECTIONNAME -src DESTINATIONSERVER -fpr FILEPATHROOT -fmt {Files and Folders,Byte Offset,Time-code,Frames,DPX} [-in SOURCEFILE] [-out DESTINATIONFILE] [-start START] [-end END] [-inst INSTANCE]
```

When there is a list of values, use pipe delimiter: -start "10|20|30" -end "100|200|300"

copy

Submits a New Instance Creation request on the media specified by mediaName to the Manager, and the Manager chooses the appropriate instance to be created. The request will fail if the requested object is on media that is not available.

Use reqinfo command to know when the request has been executed.

On success, divascript prints the request ID on the output

Synopsis

```
DivaScriptPy.exe copy -col COLLECTION [-inst INSTANCE] -grp MEDIA -obj OBJECT-NAME [-opt OPTIONS] [-prio PRIORITY]
```

copyas

Submits a request for copying an archived object to a new object, with another name or category, to the DIVA Core Manager. The Manager chooses the appropriate instance as the source of the copy. The request will fail if the requested object is on an unavailable media. All types of transfers (disk to disk, disk to tape, tape to disk, and tape to tape) are supported.

Use reqinfo command to know when the request has been executed.

On success, divascript prints the request ID on the output

Synopsis

```
DivaScriptPy.exe copyas -col COLLECTION [-com COMMENTS] [-inst INSTANCE] [-media MEDIA] -newcol NEWCOLLECTIONNAME -newmedia NEWMEDIA -newobj NEWOBJECTNAME -obj OBJECT [-opt OPTIONS] [-prio PRIORITY]
```

delete

Submits an Object Delete Request to the DIVA Core Manager. The Manager deletes the specified instance OR every instance of the object if the instance is -1.

Use reqinfo command to know when the request has been executed.

On success, divascript prints the request ID on the output

Synopsis

```
DivaScriptPy.exe delete [-col COLLECTION] [-inst INSTANCE] [-grp MEDIA] -obj OBJECTNAME [-opt OPTIONS] [-prio PRIORITY]
```

trarchive

Submits a TranscodeArchive request to the DIVA Core Manager. The original object will be restored to the local Actor cache, then transcoded to the format defined in the option field. A new object containing the transcoded clip will then be archived back to DIVA Core.

Use reqinfo command to know when the request has been executed.

On success, divascript prints the request ID on the output.

Synopsis

```
DivaScriptPy.exe trarchive [-delc CASCADEDELETE] [-com COMMENTS] -dcol DESTINATIONCOLLECTIONNAME -dobj DESTINATIONOBJECTNAME -media MEDIA [-opt OPTIONS] [-prio PRIORITY] [-qos QOS] -col SOURCECOLLECTIONNAME [-inst SOURCEINSTANCEID] -obj SOURCEOBJECTNAME
```

scan

Submits a scan tape request to the DIVA Core Manager. This call returns as soon as the Manager accepts the request.

Use reqinfo command to know when the request has been executed.

On success, divascript prints the request ID on the output.

Synopsis

```
DivaScriptPy.exe scan -col COLLECTION [-com COMMENTS] [-exl EXCLUSION] [-imm IMPORTMODE] [-incl INCLUSION] [-min MINFILESIZE] [-prio PRIORITY] -barcode BARCODE -grp MEDIA -trial TRIALMODE
```

stage

Submits a Stage request to the media specified by targetMedia.

Use reqinfo command to know when the request has been executed.

On success, DivaScriptPy prints the request ID on the output.

Synopsis

```
DivaScriptPy.exe stage -col COLLECTION -days NUMDAYS -obj OBJECT -opt OPTIONS [-prio PRIORITY] -retier RESTORETIER -stocls STORAGECLASS -target TARGETMEDIA
```

The DIVA Java API

The DIVA Java API documentation is included with the Java API installation, but you can also access it externally.

Note: Telestream has deprecated the DIVA C++ API and the DIVA Java API.

Topics

- [Viewing Java API Documentation Before Installation](#)
- [Location of DIVA Java API Documentation](#)
- [DIVA Java API Known Issues](#)

Note: Telestream strongly recommends that you use the DIVA REST API rather than previous APIs such as the DIVA Java API. The DIVA C++ API is deprecated, but supported for backward compatibility. The DIVA REST API offers new and enhanced features and security.

Viewing Java API Documentation Before Installation

The DIVA Java API documentation is available before installation, in the DIVA software download on the Telestream Software Delivery Cloud. The documentation is located in the JavaAPI directory in HTML format, and is viewable using any web browser.

Use this procedure to view the documentation before installing the API:

1. Download the DIVA Java API package located at:
https://www.telestream.net/download-files/diva/DIVArchiveAPI_V7.5.0.6.exe
2. Unzip the downloaded file and navigate to the JavaAPI directory.
3. Unzip the Java API compressed file.

4. Double-click the index.html file to open the documentation in your browser.

Location of DIVA Java API Documentation

After installing the DIVA Java API, the documentation is available in the `~/doc/JavaDoc` directory.

DIVA Java API Known Issues

Here are known issues for DIVA Java API:

- The DIVA Java API doesn't support Partial Restore Instance; it always uses -1 as the Instance ID. DIVA always picks the instance automatically.
- Older DIVA Java API releases do not return the same status codes as the DIVA C++ API. Calls that fail in the DIVA C++ API due to `DIVA_ERR_INVALID_PARAMETER` might fail in the DIVA Java API with a different error code; for example, `DIVA_ERR_INTERNAL`.
- `GetObjectsDetailsList` doesn't work correctly in the DIVA Java API 7.0 and prior releases. `GetObjectsDetailsList` terms cause random timeout and hang issues, which break the automation. Therefore, `GetObjectsDetailsList` terms are skipped during automated tests.

The DIVA C++ API

This chapter provides details about programming the DIVA C++ API.

Note: Telestream has deprecated the DIVA C++ API and the DIVA Java API.

Topics

- [The DIVA C++ API](#)
- [Session Management Commands](#)
- [Jobs and Commands](#)

Note: Telestream strongly recommends that you use the DIVA REST API rather than previous APIs such as the DIVA C++ API. The DIVA C++ API is deprecated, but supported for backward compatibility. The DIVA REST API offers new and enhanced features and security.

The DIVA C++ API

The DIVA C++ API is written in the C++ programming language. All of the definitions are contained in the include file named `DIVAapi.h`. In this document, terms in function signatures are qualified by `IN` and `OUT` to specify whether the parameter is passed as an input or an output to the function. These qualifiers are not part of the C++ language and are only used for ease of readability. You must consider that these qualifiers are equivalent to the following macro definitions:

- `#define IN`
- `#define OUT`

In this document, the term structure identifies both C-like structures and classes which have only public data members and no function members¹. Interfaces described in this document show only data members, not constructors or destructors.

The DIVA C++ API and DIVA Connect C++ API use only standard data types provided directly by the C++ language, and the vector data type provided by the Standard Template Library (STL). For more information about the vector data type, refer to the STL documentation on the OTN.

Note: The Solaris operating system doesn't support the DIVA C++ API.

DIVA does not support the following API calls and features when used with complex objects. Even if they are enabled, they will not be executed and no warnings are generated.

- `VerifyFollowingArchive`
- `VerifyFollowingRestore`
- `DeleteOnSource`
- `DeleteFile`
- `getObjectListbyFileName`
- The `getObjectInfo` and `getObjectDetailsList` will only return a single file

When copying complex objects to legacy-formatted media, the Copy job terminates returning a "Can't write a complex object in Legacy format" error, and an error code through the API.

DIVA C++ API Requirements

The DIVA C++ API supports several Microsoft Visual C++ compilers and versions of Windows Server.

Supported Compilers

The DIVA C++ API is compiled and tested using the following compilers:

- Microsoft Visual C++ 2010 (Release 10)
Including Microsoft Platform SDK 7.0a (April 2010)
- Microsoft Visual C++ 2012 (Release 11)
Including Microsoft Platform SDK 7.1A (November 2012)
- Microsoft Visual C++ 2013 (Release 13)
Including Microsoft Platform SDK 8.0A (October 2013)
- Microsoft Visual C++ 2019
Including Microsoft Platform SDK 10

1. The operators `new` and `delete` are not considered function members.

Supported Servers

There are two separate variants of the DIVA C++ API for Windows: 32-bit and 64-bit. The 32-bit model can be used on both x86 and x64 platforms. However, the 64-bit variant requires a 64-bit platform. The DIVA C++ API for Windows is supported on the following Windows releases:

- Microsoft Windows Server 2022
- Microsoft Windows Server 2019
- Microsoft Windows Server 2016

DIVA C++ API Library Options

The DIVA C++ API is delivered with both static and dynamic libraries. Each library is available in a standard format with debug support and Unicode compatibility. The different options may be found in the following build directories:

- Static Library
Static_Release
- Static Library with Debug Support
Static_Debug
- Dynamic Library
Dynamic_Release
- Dynamic Library with Debug Support
Dynamic_Debug

DIVA C++ API Compilation

Choose the 8 Bytes setting for the Strict Member Alignment option under C/C++ Code Generation in the project settings.

The following list identifies the library path that corresponds to each run time library. The run time library is normally changed automatically depending upon the selected build configuration.

- Multithreaded
Static_Release
- Debug Multithreaded
Static_Debug
- Multithreaded DLL
Dynamic_Release
- Debug Multithreaded DLL
Dynamic_Debug

The DIVA API.lib file, or the path to this file, must be included in the link settings (see [Initiator Sample Program DIVA C++ API Usage](#)). The API can be included in an application compiled with either the IDE or a script using the command line compiler.

After the application is built, either add the folder where the API.dll file is located to your PATH environment variable, or copy the API.dll file into the folder containing the executable file.

Initiator Sample Program DIVA C++ API Usage

The Initiator program is included with the API and is an example of the API usage. This is a command line program that uses the API to send jobs and get data from DIVA. Use the following project files to view the compiler settings and build the program:

- Visual C++ .NET (Release 10)
doc\CppInitiator\InitiatorVc100.vcxproj(64-bit API)
- Visual C++ .NET (Release 11)
doc\CppInitiator\InitiatorVc110.vcxproj(64-bit API)
- Visual C++ .NET (Release 12)
doc\CppInitiator\InitiatorVc120.vcxproj(64-bit API)

DIVA Release Compatibility

Features added to DIVA after the previous release of the DIVA C++ API aren't supported.

To use all DIVA features, update the client system to the latest DIVA C++ API release. The DIVA C++ API is compatible with DIVA releases 9.0 and later.

Managing Connections

The number of connections to the Manager is limited by the Manager and set in the Manager configuration file. The default configuration is two hundred connections, which includes GUI connections and all API connections. After the configured limit is reached, the API will not allow additional connections to be created. See the manager.conf file for additional information.

Caution: It is recommended that a new connection not be created for each job or command sent to the Manager. Whenever possible allow the connection to remain open for the lifetime of the session, or application.

Using the DIVA C++ API in Multithreaded Applications

The DIVA C++ API supports using multiple threads concurrently with the following restrictions (see the related function's specific documentation for additional information):

- The `DIVA_connect()` and `DIVA_disconnect()` functions share the same critical section. Although multiple simultaneous connections are supported, they must be opened and closed one at a time.
- The `init`, `get`, and `close` functions used to retrieve list information (Objects List or Objects Tape Information List) also use a Critical Section to prevent concurrent threads reinitializing the list while another thread is currently reading it. The critical section is entered when the list is initialized and left when the list is closed. There are two separate critical sections, one for each type of list.
- All of the other DIVA functions can be called simultaneously by different threads. For example, one thread can call the `DIVA_archiveObject()` function while another one is calling `DIVA_getArchiveSystemInfo()`.

DIVA C++ API Unicode Strings

The DIVA C++ API (and other DIVA components) support wide character strings. Only 64-bit Unicode is delivered with the API. The `_UNICODE` constant must be defined before including the `DIVAapi.h` header file to be able to use the `wchar_t` and `wstring`.

In addition, the application must be linked with one of the Unicode releases in the library (for example, in `lib/Release_Unicode`).

Defining, or not defining, the `_UNICODE` macro will change the implementation of the `DIVA_STRING` and `DIVA_CHAR` types.

The `_T` macro is recommended when working with static strings:

Example:

```
_T("Hello")
```

Type	_UNICODE Not Defined	_UNICODE Defined
DIVA_STRING	string	wstring
DIVA_CHAR	char	wchar_t

Session Management Commands

The following three sections describe the commands used to control the session connection.

DIVA_getApiVersion

Returns the string pointed to by version of the major part of the release number.

Synopsis

```
#include "DIVAapi.h"
```

```
void DIVA_getApiVersion (
    OUT DIVA_STRING    *version
);
```

- version

Points to a string that contains the major part of the release for this API.

DIVA_SSL_initialize

The DIVA_SSL_initialize call sets the environment for secure communication with the Manager Service. Call DIVA_SSL_initialize before calling DIVA_connect with DIVA, otherwise the DIVA_connect call will not establish a secure connection.

Synopsis

```
DIVA_STATUS DIVA_SPEC DIVA_SSL_initialize(
    DIVA_STRING KeyPath, // [in] Full path of the Key file contain
    the private key and certificate in PEM format.
    DIVA_STRING TrustStorePath, // [in] Full path of the file
    containing Trust certificates in PEM format.
    DIVA_STRING KeyPassword // [in] Password for the private key
)
```

DIVA_connect

Opens a connection with the Manager. All of the other API functions are only available when a connection is open. A connection cannot be opened if another connection is already open. To open a new connection, the previous one must be explicitly closed by calling DIVA_disconnect().

Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_connect (
    IN string managerAddress,
    IN int portNumber
);
DIVA_STATUS DIVA_connect (
    IN string managerAddress,
    IN int portNumber,
    IN string userName,
    IN string password,
    IN string applicationName
);
DIVA_STATUS DIVA_connect (
    IN string managerAddress,
    IN int portNumber,
    IN string userName,
    IN string password,
    IN string applicationName
    IN string userInfo
);
```

- `managerAddress`
The IP address of the Manager.
- `portNumber`
The Manager listening port. The default port is pointed to by the constant value `DIVA_MGER_DEFAULT_PORT`.
- `userName`
The user name.
- `password`
The password associated with the user name.
- `applicationName`
The name of the application.
- `userInfo`
User specific and specified information.

Multithreaded Applications:

A critical section protects both the `DIVA_connect()` and `DIVA_disconnect()` functions. If a thread is already in the process of closing the connection to the Manager, other threads must wait until the running thread exits the `DIVA_connect()` function before being able to open or close the connection.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system is no longer able to accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.

- `DIVA_ERR_NO_ARCHIVE_SYSTEM`
There was a problem when establishing a connection with the specified DIVA system.
- `DIVA_ERR_WRONG_VERSION`
The release levels of the API and the Manager are not compatible.
- `DIVA_ERR_ALREADY_CONNECTED`
A connection is already open.

Also see [DIVA_disconnect](#).

DIVA_disconnect

Closes a connection with the Manager. When a connection is closed, only the `DIVA_connect()` function can be called. If no connection is currently open, this function has no effect and returns `DIVA_OK`.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_disconnect ()
```

Multithreaded Applications

A critical section protects both the `DIVA_connect()` and `DIVA_disconnect()` functions. If a thread is already in the process of closing the connection to the Manager, other threads must wait until the running thread exits the `DIVA_disconnect()` function before being able to open or close the connection.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.

- `DIVA_ERR_DISCONNECTING`

There was a problem when disconnecting. The connection is considered to still be open.

Also see [DIVA_connect](#).

Jobs and Commands

The following sections discuss all of the available DIVA C++ API commands for use in an application.

DIVA_addGroup

This function adds a new Tape Group.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_addGroup (
IN  DIVA_STRING      groupName,
IN  int              associatedSet,
IN  DIVA_STRING      comment,
IN  bool             toBeRepacked,
IN  bool             worstFitEnabled,
IN  int              worstFitRepackTapes,
IN  int              mediaFormatId
);
```

- `groupName`

The name of the Tape Group to be added.

- `associatedSet`

The set of tapes to associate with the new Tape Group. This value must be strictly greater than zero.

- `comment`

A text description of the new Tape Group.

- `toBeRepacked`

If true, tapes belonging to this Tape Group are eligible for automatic repacking.

- `worstFitEnabled`

If true, Worst Fit Policy (access speed optimization) will apply.

- `worstFitRepackTapes`

The number of tapes reserved for Worst Fit Repacking.

- `mediaFormatId`

The data format to be used by the tapes assigned to this Tape Group. The value can be `DIVA_MEDIA_FORMAT_LEGACY` or `DIVA_MEDIA_FORMAT_AXF`.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
DIVA system can not accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set using the `DIVA_API_TIMEOUT` variable. The default value is one hundred-eighty (180) seconds.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_GROUP_ALREADY_EXISTS`
The specified Tape Group already exists.

DIVA_archiveObject

Submits an archive job to the Manager. This function returns as soon as the Manager accepts the job. The application must call the function `DIVA_getRequestInfo()` to check that the operation completed successfully.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_archiveObject (
IN DIVA_STRING          objectName,
IN DIVA_STRING          objectCollection,
IN DIVA_STRING          source,
IN DIVA_STRING          mediaName,
IN DIVA_STRING          filePathRoot,
IN vector<DIVA_STRING>   filenamesList,
IN DIVA_ARCHIVE_QOS     qualityOfService,
IN int                  priorityLevel,
IN DIVA_STRING          comments,
```

```
IN DIVA_STRING      archiveOptions,
OUT int            requestNumber
);
```

- `objectName`

The name of the object to be archived.

- `objectCategory`

The Collection of the object to be archived.

- `source`

The name of the Source Server (for example, the video server, browsing server, and so on). This name must be known to the DIVA configuration description.

- `mediaName`

The tape group or disk array where the object is to be saved. The media may be defined as follows:

- `Name` (of the Tape Group or Array)

Provide the tape group or disk array name as defined in the configuration. The object is saved to the specified media and assigned to the default SP (Storage Plan).

- `SP Name`

Provide a SP Name (Storage Plan Name) as defined in the configuration. The object will be assigned to the specified Storage Plan and saved to the default media specified.

- Both of the above (Name and SP Name)

The object is saved to the specified media as in Name, and assigned to the specified Storage Plan as in SP Name. The Name and the SP Name must be separated by the & delimiter (this is configurable).

When this parameter is a null string, the default group of tapes called DEFAULT is used. Complex objects can only be saved to AXF media types.

- `filePathRoot`

The root folder for the files specified by the `filenamesList` parameter.

- `filenamesList`

List of file path names relative to the folder specified by the `filePathRoot` parameter. Path names must be absolute names when the `filePathRoot` is null.

If the `-gcinfilelist` option is specified the Genuine Checksum is included with a colon separator between the file name and the GC value as follows:

```
test1.txt:a6f62b73f5a9bf380d32f062f2d71cbc
test2.txt:96bf41e4600666ff69fc908575c0319
```

- `qualityOfService`

One of the following codes executes the job using the specified QOS:

- `DIVA_QOS_DEFAULT`

Archiving is performed according to the default Quality Of Service (currently direct and cache for archive operations).

- `DIVA_QOS_CACHE_ONLY`

Use cache archive only.

- `DIVA_QOS_DIRECT_ONLY`

Use direct archive only; no disk instance is created.

- `DIVA_QOS_CACHE_AND_DIRECT`

Use cache archive if available, or direct archive if cache archive is not available.

- `DIVA_QOS_DIRECT_AND_CACHE`

Use direct archive if available, or cache archive if direct 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`

Delete source files when the tape migration is done. Available for local Source Servers, disk Source Servers, and standard FTP Source Servers. This feature is not available for complex objects.

- `priorityLevel`

The priority level for this job. The `priorityLevel` can be in the range zero to one hundred, or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred, or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `comments`

Optional information describing the object. This can be a null string.

- `archiveOptions`

Additional options for performing the transfer of data from the Source Server to DIVA. These options supersede any options specified in the DIVA configuration database. Currently the possible values for `archiveOptions` are as follows:

- `Null string`

A null string specifies no options.

- `-delete_on_source`

Executes a delete on the Source Server after an archive job completes.

- `-r`

Using `-r` specifies that every name in `filenamesList` that refers to a folder must be scanned recursively. This also applies when `FilePathRoot` is specified and an asterisk designates the files to be archived. This option can be used when archiving from a local Source Server or from a standard FTP Server.

- `-login`

A user name and password is required to log in to some Source Servers. This option obsoletes the `-gateway` option from earlier releases.

- `-pass`

The password used with `-login`.

- `-gcinfilelist [gcType]`

Specifies that GC (Genuine Checksum) values are included in the file names list. The value of `gcType` must match the Manager's default checksum type as specified in the DIVA configuration (MD5 by default). The GC values are then used to verify the transfer from the Source Server.

- `requestNumber`

The job number assigned to this job. This number is used for querying the status or canceling the job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

Manager can no longer accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set using the `DIVA_API_TIMEOUT` variable. The default value is one hundred-eighty (180) seconds.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
The Manager or API detected an internal error.
- `DIVA_ERR_INVALID_PARAMETER`
The Manager did not understand a parameter value.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is three hundred.
- `DIVA_ERR_GROUP_DOESNT_EXIST`
The specified tape group or disk array does not exist.
- `DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST`
The specified Server is unknown by the DIVA system.

DIVA_associativeCopy

Submits a job for creating new instances in the Tape Group (specified by group). DIVA guarantees that these instances are stored sequentially on tapes:

- The job is completed only when every object is copied to the same tape.
- In the case of drive or tape failure during a write operation, instances currently written are erased and the job is retried once.
- The choice of the tape to be used for the copy follows the policy used for the archive operation (written tapes with enough remaining size regardless of optimizations).
- Associative Copy does not span tapes—the job terminates (and is retried once) instead of spanning. The job terminates if the sum of the size of the objects to copy exceeds the capacity of every individual tape present in the Managed Storage.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_associativeCopy (
IN vector<DIVA_OBJECT_SUMMARY>    *objectsInfo,
IN DIVA_STRING                     groupName,
IN int                             priorityLevel,
IN DIVA_STRING                     options,
OUT int                            *requestNumber
);
```

- `objectsInfo`

A pointer to a list of objects defined by a name and Collection pair.

- `groupName`

The name of the Tape Group where the new instance will be located. Complex objects can only be saved to AXF media types. Associative Copy to a disk array is not available.

- `priorityLevel`

The level of priority for this job. The `priorityLevel` can be in the range zero to one hundred or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred is the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `options`

An optional string attribute for specifying additional terms to the job.

- `requestNumber`

A number identifying the job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The Manager system is no longer able to accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

- `DIVA_ERR_INVALID_PARAMETER`

A parameter value was not understood by the Manager.

- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`

The count of simultaneous jobs reached the maximum allowed value. This variable is set in the `manager.conf` configuration file and the default value is three hundred.

- `DIVA_ERR_OBJECT_DOESNT_EXIST`

The specified object does not exist in the database.

- `DIVA_ERR_SEVERAL_OBJECTS`

More than one object with the specified name exists in the database.

- `DIVA_ERR_OBJECT_OFFLINE`

No available instance for this object. Tape instances are ejected and no Actor could provide a disk instance.

- `DIVA_ERR_GROUP_DOESNT_EXIST`

The specified tape group or disk array does not exist.

- `DIVA_ERR_OBJECT_IN_USE`

The object is currently in use (being archived, restored, deleted, and so on).

- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`

The specified object has instances that are partially deleted.

Also see [DIVA_archiveObject](#) and [DIVA_copyToGroup and DIVA_copy](#).

DIVA_cancelRequest

Submits a Cancel operation to the Manager. This function returns as soon as the Manager accepts the operation. The application must call the function `DIVA_getRequestInfo()` to check that the operation was successful.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_cancelRequest (
IN int      requestNumber,
IN DIVA_STRING options
```



```
);
```

- `requestNumber`

A number identifying the job to be canceled. This parameter can be set to `DIVA_ALL_REQUESTS` to cancel all cancel-able jobs.

- `options`

An optional string attribute for specifying additional terms to the job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

- `DIVA_ERR_NO_SUCH_REQUEST`

The `requestNumber` identifies no job.

Also see [DIVA_getRequestInfo](#).

DIVA_changeRequestPriority

Submits a Change Request Priority job to the Manager. This function returns as soon as the Manager accepts the job. The application must call the `DIVA_getRequestInfo()` function to check that the operation was successful.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_changeRequestPriority (
IN int      requestNumber,
```

```
IN int          priorityLevel,
IN DIVA_STRING  passThruOptions
);
```

- `requestNumber`

A number identifying the job to be changed.

- `priorityLevel`

The level of priority for this job. The `priorityLevel` can be in the range zero to one hundred. The value zero is the lowest priority and one hundred is the highest priority.

There are five predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`

The use of `DIVA_DEFAULT_REQUEST_PRIORITY` is not allowed with this function.

Using a value either outside of the range of zero to one hundred or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `passThruOptions`

An optional string attribute for specifying additional terms to the job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

- `DIVA_ERR_NO_SUCH_REQUEST`
The requestNumber identifies no job.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value has not been understood by the Manager.

Also see [DIVA_getRequestInfo](#).

DIVA_copyToGroup and DIVA_copy

Submits a new instance creation job on the media specified by `mediaName` to the Manager, and the Manager chooses the appropriate instance to be created. This function returns as soon as the Manager accepts the job. The application must call the `DIVA_getRequestInfo()` function to check that the operation was successful.

The job will fail if the job's object is on media that is not available. The Media Names (tape barcodes and disk names) that contain instances of the object will be included in the `additionalInfo` field of the `DIVA_getRequestInfo()` response.

A tape group may contain two instances of the same object. In this case, Manager will terminate the job if both instances cannot be written on two different tapes. A disk array can contain two instances of the same object; however Manager will terminate the job if the new instance cannot be written on a different disk. There can be a maximum of only one instance of each object per disk or tape.

Synopsis

`DIVA_copyToGroup` is a public alias to `DIVA_copy` and performs the same functionality.

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_copy (
IN  DIVA_STRING      objectName,
IN  DIVA_STRING      categoryName,
IN  int              instanceID,
IN  DIVA_STRING      mediaName,
IN  int              priorityLevel,
IN  DIVA_STRING      options,
OUT int              *requestNumber
);
```

```
DIVA_STATUS DIVA_copyToGroup (
IN  DIVA_STRING      objectName,
IN  DIVA_STRING      categoryName,
IN  int              instanceID,
IN  DIVA_STRING      mediaName,
IN  int              priorityLevel,
IN  DIVA_STRING      options,
OUT int              *requestNumber
);
```

- `objectName`
The name of the object to be copied.
- `objectCategory`
The Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.
- `instanceID`
The instance's identifier. `DIVA_ANY_INSTANCE` as the Instance ID means that DIVA will choose the appropriate instance.
- `mediaName`
The media (tape group or disk array) where the new instance will be located.
- `priorityLevel`
The level of priority for this job. The `priorityLevel` can be in the range zero to one hundred or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred is the highest priority.
There are six predefined values as follows:
 - `DIVA_REQUEST_PRIORITY_MIN`
 - `DIVA_REQUEST_PRIORITY_LOW`
 - `DIVA_REQUEST_PRIORITY_NORMAL`
 - `DIVA_REQUEST_PRIORITY_HIGH`
 - `DIVA_REQUEST_PRIORITY_MAX`
 - `DIVA_DEFAULT_REQUEST_PRIORITY`
When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.
Using a value either outside of the range of zero to one hundred or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.
- `options`
An optional string attribute for specifying additional terms to the job.
- `requestNumber`
A number identifying the job to be changed.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the *Manager*.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

- `DIVA_ERR_INVALID_PARAMETER`

A parameter value has not been understood by the Manager.

- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`

The count of simultaneous jobs has reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default is three hundred.

- `DIVA_ERR_OBJECT_DOESNT_EXIST`

The specified object does not exist in the database.

- `DIVA_ERR_INSTANCE_DOESNT_EXIST`

The instance specified for restoring this object does not exist.

- `DIVA_ERR_SEVERAL_OBJECTS`

More than one object with the specified name exists in the database.

- `DIVA_ERR_OBJECT_OFFLINE`

No available instance for this object. Tape instances are ejected and no Actor could provide a disk instance.

- `DIVA_ERR_INSTANCE_OFFLINE`

The instance specified for restoring this object is ejected, or the Actor owning the specified disk instance is not available.

- `DIVA_ERR_GROUP_DOESNT_EXIST`

The specified Tape Group does not exist.

- `DIVA_ERR_OBJECT_IN_USE`

The object is currently in use (being archived, restored, deleted, and so on).

- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`

The specified object has instances that are partially deleted.

Also see [DIVA_archiveObject](#).

DIVA_copyToNewObject

Submits a job for copying an archived object to a new object, with another name or Collection, to the Manager. The Manager chooses the appropriate instance as the source of the copy. This function returns as soon as the Manager accepts the job. The application must call the `DIVA_getRequestInfo()` function to check that the operation was successful.

The job will fail if the job's object is on an unavailable media. The media names (tape barcodes and disk names) that contain instances of the object will be included in the `additionalInfo` field of the `DIVA_getRequestInfo()` response.

All types of transfers (disk to disk, disk to tape, tape to disk, and tape to tape) are supported.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_copyToNewObject (
IN const DIVA::ObjectInstanceDescriptor    &source,
IN const DIVA::ObjectInstanceDescriptor    &target,
IN const DIVA::RequestAttributes            &attrs,
IN DIVA_STRING                             options,
OUT int                                    *requestNumber
);
```

- `source`

The description of the object or object instance to be copied:

- `source.objectName`

The Source Server object name (required).

Note: Object Names cannot begin with a dollar sign (\$).

- `source.objectCategory`

The Source Server object Collection (required).

- `source.group`

The Source Server object instance tape group or disk array. This is optional. However, if specified DIVA will use this instance as the Source Server.

- `source.instanceID`

The Instance ID of the Source Server object instance. This is optional. However, if specified and not equal to `DIVA_ANY_INSTANCE`, DIVA will use this instance as the Source Server. The `source.group` parameter will be ignored if `source.instanceID` is specified.

If both `source.group` and `source.instanceID` are omitted, DIVA will use the most suitable instance (that provides the best performance) as a source.

- `target`

The description of the target object:

- `target.objectName`
The target object name (required).

Note: Object Names cannot begin with a dollar sign (\$).

- `target.objectCategory`
The target object Collection (required).
- `target.group`
See the following paragraph.
- `target.instanceID`
This call ignores this value.
Either the object name or Collection (or both) must be different from name or Collection of the Source Server object. The job will fail if the target object already exists in DIVA.
- `attrs`
The job attributes:
- `attrs.priority`
The job priority (optional). If this is not explicitly set the default value is used. Possible values are zero (lowest) to one hundred (highest).
- `attrs.qos`
QOS (Quality of Service) is not applicable to this job and this call ignores this value.
- `attrs.comments`
The target object's comments (optional). If no value is specified the Source Server object's comments are inherited.
- `attrs.options`
This job has no additional options and this call ignores this value.
- `requestNumber`
The number identifying the job that is returned by DIVA.

```

DIVA_STATUS DIVA_copyToNewObject (
IN const DIVA_STRING      &objectName,
IN const DIVA_STRING      &objectCategory,
IN const DIVA_STRING      &objectMedia,
IN int                    objectInstanceID,
IN const DIVA_STRING      &newObjectName,
IN const DIVA_STRING      &newObjectCategory,
IN const DIVA_STRING      &newObjectInstanceMedia,
IN const DIVA_STRING      &comments,
IN int                    priorityLevel,
IN DIVA_STRING            options,
OUT int                   *requestNumber
);

```

- `objectName`
The name of the Source Server object.
- `objectCategory`
The Collection of the Source Server object.
- `objectMedia`
The tape group or disk array of the Source Server object instance (optional). If specified (not empty), DIVA will use this instance as a Source Server. Complex objects can only be saved to AXF formatted media types.
- `objectInstanceID`
The Instance ID of the Source Server object instance (optional). If specified and not equal to `DIVA_ANY_INSTANCE`, DIVA will use this instance as the Source Server. This call ignores the `ObjectMedia` parameter if an `instanceID` value is specified. If both `objectMedia` and `instanceID` are not specified, DIVA will use the most suitable instance (providing the best performance) as the Source Server.
- `newObjectName`
The target object name.

Note: Object Names cannot begin with a dollar sign (\$).

- `newObjectCategory`
The target object Collection. Either the object name or Collection (or both) must be different from name or Collection of the Source Server object. This job will fail if the target object already exists in DIVA.
- `newObjectInstanceMedia`
The tape group or disk array where the object will be saved. The media may be defined as follows:
- Name (of the Tape Group or Array)
Provide the tape group or disk array name as defined in the configuration. The object is saved to the specified media and assigned to the default SP (Storage Plan).

- `SP Name`
Provide a Storage Plan Name as defined in the configuration. The object will be saved to the default media specified in the Storage Plan and assigned to the specified Storage Plan.
- Both of the above (Name and SP Name)
The object is saved to the specified media as in Name above. The object is assigned to the specified SP as in SP Name above. The Name and the SP Name must be separated by the & delimiter (this is configurable).
- `comments`
Optional information describing the target object. If left empty the Source Server object comments are inherited.
- `priorityLevel`
Level of priority for this job. The possible values can be in the range zero to one hundred, and the `DIVA_DEFAULT_REQUEST_PRIORITY` (use default job priority).
- `options`
Optional string attribute for specifying additional terms to the job.
- `requestNumber`
The job number assigned to this job by DIVA.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.

- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs has reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is three hundred.
- `DIVA_ERR_OBJECT_DOESNT_EXIST`
The specified object does not exist in the database.
- `DIVA_ERR_INSTANCE_DOESNT_EXIST`
The instance specified for restoring this object does not exist.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.
- `DIVA_ERR_OBJECT_OFFLINE`
No available instance for this object. Tape instances are ejected and no Actor could provide a disk instance.
- `DIVA_ERR_INSTANCE_OFFLINE`
The instance specified for restoring this object is ejected, or the Actor owning the specified disk instance is not available.
- `DIVA_ERR_GROUP_DOESNT_EXIST`
The specified Tape Group does not exist.
- `DIVA_ERR_OBJECT_IN_USE`
The object is currently in use (being archived, restored, deleted, and so on).
- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`
The specified object has instances that are partially deleted.

Also see [DIVA_copyToGroup](#) and [DIVA_copy](#).

DIVA_deleteGroup

Deletes the Tape Group passed as an argument. A Tape Group can only be deleted when the Tape Group is empty.

Synopsis

```
#include "DIVAapi.h"
```

```
IN DIVA_STRING          groupName
DIVA_STATUS DIVA_deleteGroup (
);
```

- `groupName`

The name of the Tape Group to be deleted.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`.

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_GROUP_DOESNT_EXIST`
The specified Tape Group does not exist.
- `DIVA_ERR_GROUP_IN_USE`
The Tape Group contains at least one object currently in use (being archived, restored, deleted, and so on).

DIVA_deleteInstance

Deletes an object instance.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_deleteInstance (  
IN DIVA_STRING      objectName,  
IN DIVA_STRING      categoryName,  
IN int              instanceID,  
IN int              priorityLevel,  
IN DIVA_STRING      options,  
OUT int              *requestNumber  
);
```

```

DIVA_STATUS DIVA_deleteInstance (
IN DIVA_STRING      objectName,
IN DIVA_STRING      categoryName,
IN DIVA_STRING      mediaName,
IN int              priorityLevel,
IN DIVA_STRING      options,
OUT int             *requestNumber
);

```

- objectName

The name of the object to be deleted.

- objectCategory

The Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.

- instanceID

The instance's identifier

- mediaName

Defines the media that contains the valid instance. If the instanceID is -1, the instance on the media will be deleted. If the media contains two or more instances, only one of the instances will be deleted.

- priorityLevel

The level of priority for this job. The priorityLevel can be in the range zero to one hundred or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred is the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- options

An optional string attribute for specifying additional terms to the job.

- requestNumber

A number identifying the job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs has reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is three hundred.
- `DIVA_ERR_OBJECT_DOESNT_EXIST`
The specified object does not exist in the database.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.
- `DIVA_ERR_INSTANCE_DOESNT_EXIST`
The specified instance does not exist.
- `DIVA_ERR_LAST_INSTANCE`
`DIVA_deleteObject()` must be used to delete the last instance of an object.
- `DIVA_ERR_OBJECT_IN_USE`
The object is currently in use (being archived, restored, deleted, and so on).
- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`
The specified object has instances that are partially deleted.

See also [DIVA_getObjectInfo](#).

DIVA_deleteObject

Submits an object Delete job to the Manager. The Manager deletes every instance of the object. This function returns as soon as the Manager accepts the job. To check that the operation was successful the application must call the function

`DIVA_getRequestInfo()`.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_deleteObject (
IN  DIVA_STRING      objectName,
IN  DIVA_STRING      objectCategory,
IN  int              priorityLevel,
IN  DIVA_STRING      options,
OUT int              *requestNumber
);
```

- `objectName`

The name of the object to be deleted.

- `objectCategory`

The Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.

- `priorityLevel`

The level of priority for this job. The `priorityLevel` can be in the range zero to one hundred or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred is the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `options`

An optional string attribute for specifying additional terms to the job.

- `requestNumber`
A number identifying the job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs has reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is three hundred.
- `DIVA_ERR_OBJECT_DOESNT_EXIST`
The specified object does not exist in the database.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.
- `DIVA_ERR_OBJECT_IN_USE`
The object is currently in use (being archived, restored, deleted, and so on).
- `DIVA_ERR_OBJECT_BEING_ARCHIVED`
The specified object does not exist in the database, but it is currently being archived.

See also [DIVA_getRequestInfo](#) and [DIVA_deleteInstance](#).

DIVA_ejectTape

Submits an Eject job to DIVA. The job completes when the specified tapes are outside of the Managed Storage.

If at least one of the tapes does not exist, is already ejected, or currently in use by another job, the `DIVA_ERR_INVALID_PARAMETER` status code is returned and no tapes are ejected.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_ejectTape (
IN vector<DIVA_STRING>    *vsList,
IN bool                    release
IN DIVA_STRING            comment,
IN int                    priorityLevel,
OUT int                   *requestNumber
);
```

- `vsList`

List of VSNs for identifying the tapes to be ejected.

- **release**

When true, perform a `DIVA_release()` on every instance located on the successfully ejected tapes.

- `comment`

Externalization comment.

- `priorityLevel`

The level of priority for this job. The `priorityLevel` can be in the range zero to one hundred or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred is the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `requestNumber`

The number identifying the job.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the *Manager*.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager, or at least one of the bar-codes refers to a bad tape (that is, an unknown tape, offline tape, or tape in use).
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs has reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is three hundred.

See also [DIVA_insertTape](#).

DIVA_enable_Automatic_Repack

Enable or disable the automatic repack scheduling in the Manager.

When the automatic repack scheduling is enabled, the schedule defined in the Web App is applied and tapes belonging to Tape Groups for which repack is allowed can be repacked according to the other automatic repack settings.

When the automatic repack scheduling is disabled, all running automatic repack jobs might be canceled (or not, according to other automatic repack settings), and no other automatic repack jobs will be started until the automatic repack scheduling is turned on again (either from this API or from the Web App).

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_enableAutomaticRepack (
IN bool      enable
);
```

- enable

Set true to enable automatic repack scheduling, false to disable.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`.

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the *Manager*.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.

DIVA_getArchiveSystemInfo

Retrieves general information about the DIVA system.

A DIVA system communicates with a Robotic System composed of one or more independent ACSs (Automated Cartridge Systems). An ACS is composed of one or more LSMs (Managed Storage Modules) that can exchange tapes through a PTP (Pass Through Port). Each tape drive is located in a LSM.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getArchiveSystemInfo (
IN string      options;
OUT DIVA_GENERAL_INFO *info
);
```

- info

Pointer to a `DIVA_GENERAL_INFO` structure that will be modified to include information about the DIVA system.

```
typedef enum {
    DIVA_IS_ON = 0,
    DIVA_IS_OFF,
    DIVA_GLOBAL_STATE_IS_UNKNOWN
} DIVA_GLOBAL_STATE;

typedef enum {
    DIVA_LIBRARY_OK = 0,
    DIVA_LIBRARY_OUT_OF_ORDER,
    DIVA_LIBRARY_STATE_UNKNOWN
} DIVA_LIBRARY_STATE;

class DIVA_ACTOR_AND_DRIVES_DESC {
public:
    string                actorName;
    string                actorAddress;
    bool                  actorIsAvailable;
    vector<string>        *connectedDrives;
    bool                  repackEnabled;
    bool                  classicEnabled;
    bool                  cacheArchiveEnabled;
    bool                  directArchiveEnabled;
    bool                  cacheRestoreEnabled;
    bool                  directRestoreEnabled;
    bool                  deleteEnabled;
    bool                  copyToGroupEnabled;
    bool                  associativeCopyEnabled;
    int                   cacheForRepack;
};

class DIVA_LSM_DESC {
public:
    string                lsmName;
    int                   lsmID;
    bool                  lsmIsAvailable;
};

class DIVA_DRIVE_DESC {
public:
    string                driveName;
    int                   driveTypeID;
    string                driveType;
    int                   lsmID;
    bool                  driveIsAvailable;
    bool                  repackEnabled;
    bool                  classicEnabled;
};

class DIVA_GENERAL_INFO {
public:
    DIVA_GLOBAL_STATE status;
    DIVA_LIBRARY_STATE lib_status;
```

```

int
vector<DIVA_ACTOR_AND_DRIVES_DESC>
vector<DIVA_LSM_DESC>
vector<DIVA_DRIVE_DESC>
int
long
long
int
vector<int>
vector<int>
int
int
int
string
string
int
int
int
int
totalNumberOfObjects;
*actorsDrivesList;
*lsmlist;
*drivesList;
numberOfBlankTapes;
remainSizeOnTapes;
totalSizeOnTapes;
capSize;
*pendingRequests;
*currentRequests;
numOfAvailableActors
numOfAvailableDrives
numOfAvailableDisks
siteName
siteIpAddress
sitePort
firstUsedRequestId
lastUsedRequestId
};

```

The following terms are listed in the order they appear in the preceding code example. Therefore there may be duplicates because the same parameter is used in different places in the code to represent different items.

- actorName
The name of the Actor.
- actorAddress
The Actor IP address.
- actorIsAvailable
Determines if the Actor is available.
- connectedDrives
Identifies the connected drives.
- repackEnabled
This is true if Repack is enabled.
- classicEnabled
This parameter is maintained for compatibility purposes only. This is only true if all seven standard operations are enabled.
- cacheArchiveEnabled
This is true if Cached Archive is enabled.
- directArchiveEnabled
This is true if Direct Archive is enabled.
- cacheRestoreEnabled
This is true if Cached Restore is enabled.
- directRestoreEnabled
This is true if Direct Restore is enabled.

- `deleteEnabled`
This is true if Delete is enabled.
- `copyToGroupEnabled`
This is true if Copy To Group is enabled.
- `associativeCopyEnabled`
This is true if Associative Copy is enabled.
- `cacheForRepack`
This is true if Cached Repack is enabled.
- `lsmName`
User-friendly Managed Storage Module name.
- `lsmID`
This is the unique LSM ID.
- `lsmIsAvailable`
This is true if the LSM identified by the preceding [lsmID](#) parameter is available for DIVA.
- `driveName`
This is the Drive Name.
- `driveTypeID`
This is the Drive Type ID.
- `driveType`
This is the Drive Type Name.
- `lsmID`
This is the ID of the LSM containing the drive. See [lsmList](#).
- `driveIsAvailable`
This is true if the identified drive is available for DIVA.
- `status`
The status of DIVA.
- `lib_status`
This is ok if at least one ACS is online. See [lsmList](#).
- `totalNumberOfObjects`
The number of objects managed by this DIVA system.
- `actorsDrivesList`
<DIVA_ACTOR_AND_DRIVES_DESC>
- `lsmList`
<DIVA_LSM_DESC>

- `drivesList`
`<DIVA_DRIVE_DESC>`
- `numberOfBlankTapes`
 The number of blank tapes in a Set associated with at least one Tape Group. Tape(s) may be externalized or write disabled.
- `remainSizeOnTapes`
 The sum of the remaining size of tapes (in gigabytes) that are online, in a Set associated with at least one Tape Group in an ACS where DIVA has a drive that is writable, and the remaining size on disks accepting permanent storage. Only disks that are currently visible are used in the calculation.
`Remaining_Size_of_Online_Tapes + Remaining_Size_of_Disks_Accepting_Permanent_Storage`
- `totalSizeOnTapes`
 The sum of the total size of all tapes (in gigabytes) in a Set associated with at least one Tape Group available for DIVA, and of the total size of all disks accepting storage. Only disks that are currently visible are used in the calculation.
`Total_Size_of_all_Available_Tapes + Total_Size_of_all_Disks_Accepting_Storage`
- `capSize`
 The number of slots in the default CAP.
- `pendingRequests`
 The number of pending jobs.
- `currentRequests`
 The number of current jobs.
- `numOfAvailableActors`
 The number of currently running Actors.
- `numOfAvailableDrives`
 The number of drives currently in online status.
- `numOfAvailableDisks`
 The number of disks currently in online status.
- `siteName`
 The name of the main site as entered in the Web App.
- `siteIpAddress`
 The Manager IP Address.
- `sitePort`
 The port number where the Manager is listening.
- `firstUsedRequestId`
 The first job ID used by the current Manager session. This value is -1 if no jobs were processed.

- `lastUsedRequestId`

The last job ID used by the current Manager session. This value is -1 if no jobs were processed.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

DIVA_getArrayList

The purpose of this function is to provide a list of arrays and disks associated with the arrays in the DIVA system. It also returns arrays without any disks associated with them. In DIVA 9.0 and later the Source Media Priority and storage options are reported in the returned data from this call.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getArrayList (
IN                                     string options;
OUT vector<DIVA_ARRAY_DESC>         *&arraysInfo
);
```

- `arraysInfo`

A pointer to a list of `DIVA_ARRAY_DESC` structures.

```
#ifndef WIN32
typedef long long __int64;
#endif
```

```
typedef enum {
    DIVA_CLOUD_STORAGECLASS_NONE=0,
    DIVA_CLOUD_STORAGECLASS_ARCHIVE,
    DIVA_CLOUD_STORAGECLASS_STANDARD
} DIVA_CLOUD_STORAGECLASS;

class DIVA_ARRAY_DESC {
public:
    DIVA_STRING          arrayDesc;
    DIVA_STRING          arrayName;
    int                  number_Of_Disk;
    int                  mediaFormatId;
    DIVA_CLOUD_STORAGECLASS cloudStorageClass; (deprecated)
    vector<DIVA_DISK_ARRAY> *arrayDiskList;
    DIVA_STRING          storageOptions
};

typedef enum {
    DIVA_DISK_STATUS_UNKNOWN = 0,
    DIVA_DISK_STATUS_ONLINE,
    DIVA_DISK_STATUS_OFFLINE,
    DIVA_DISK_STATUS_NOT_VISIBLE
} DIVA_DISK_STATUS;

class DIVA_DISK_ARRAY {
public:
    __int64              disk_CurrentRemainingSize;
    bool                disk_isWritable;
    __int64              disk_maxThroughput;
    __int64              disk_minFreeSpace;
    DIVA_STRING          disk_name;
    DIVA_STRING          disk_site;
    DIVA_DISK_STATUS     disk_status;
    __int64              disk_total_size;
    __int64              consumedSize;
    DIVA_STRING          disk_array_name;
};
```

- arrayDesc
The description of the array.
- arrayName
The name of the array.
- numberOfDisk
The number of disks in the array.
- mediaFormatId
The format of the data on disks in this array. The value can be DIVA_MEDIA_FORMAT_LEGACY, DIVA_MEDIA_FORMAT_AXF, or DIVA_MEDIA_FORMAT_AXF_10.

- `storageOptions`

The Storage Class and Storage Location. Formatted as follows:

- `oracle_storage_class=[NONE|ARCHIVE|STANDARD]`
- `storage_location=[LOCAL|OPC|OCI]`

- `arrayDiskList`

A list of the disks in an array.

- `DIVA_DISK_STATUS_UNKNOWN = 0`

The disk status is unknown.

- `DIVA_DISK_STATUS_ONLINE`

The disk status is online.

- `DIVA_DISK_STATUS_OFFLINE`

The disk status is offline.

- `DIVA_DISK_STATUS_NOT_VISIBLE`

The disk status is not visible.

- `disk_CurrentRemainingSize`

The current remaining disk size.

- `disk_consumedSize`

The current consumed size on disk in kilobytes. Useful for unlimited cloud disks to determine the space consumed on the disk.

- `disk_isWritable`

This flag checks to see whether the disk is writable.

- `disk_maxThroughput`

The maximum throughput of a disk.

- `disk_minFreeSpace`

The minimum free space available on a disk.

- `disk_name`

The name of the disk.

- `disk_site`

The name of the site where the disk is located.

- `disk_status`

The current disk status.

- `disk_total_size`

The total size of the disk.

- `disk_array_name`

The name of the array containing the disk.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.

DIVA_getFinishedRequestList

Get all of the finished jobs starting from the specified number of seconds before the present. Finished jobs are jobs that have completed normally or were terminated.

Use this function as follows:

If the list of jobs to be processed is greater than the batch size, make successive calls to this function. The first time the function is called, set `initialTime` to the desired number of seconds earlier, where the list is to start. The maximum is three days. For successive calls set `initialTime` to zero and set the `uniqueId` to the value returned by the previous call. The returned list will be empty after all of the jobs have been returned.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getFinishedRequestList (
IN int                                     batchSize,
IN int                                     initialTime,
IN DIVA_STRING                            uniqueId,
OUT DIVA_FINISHED_REQUEST_INFO            *pFinishedRequestInfo
);
```

- `batchSize`

The maximum size of the returned list of objects. This must be set to a value no greater than 1000; the recommended setting is 500. This is only a suggestion and may be overridden by the underlying functionality. This parameter should not be used to guarantee that the list will be a certain size.

- `initialTime`

The first time the function is called this value defines how far back in time to go to look for finished jobs. Jobs that have finished between this time and the present will be retrieved. The valid range for this parameter is 1 to 259200 (three days). If the number of jobs to be returned is greater than the batch size, the call is repeated. For these calls this parameter should be set to zero (0).

- `uniqueId`

The first time the function is called this value must be set to an empty string (`_T("")`). Do not set this parameter to NULL. If the number of job to be returned is greater than the batch size, the call is repeated. For these calls this value should be set to the `uniqueId` as found in `DIVA_FINISHED_REQUEST_INFO` that was returned by the previous call.

- `pFinishedRequestInfo`

This is a pointer to the returned data. See the description of `DIVA_FINISHED_REQUEST_INFO` later in this section. It is the user's responsibility to allocate and delete instances of this class.

```
class DIVA_FINISHED_REQUEST_INFO {
public:
    DIVA_STRING                uniqueId;
    vector<DIVA_REQUEST_INFO> *pRequestList;
};
```

- `uniqueId`

After the first (and any subsequent) call, the API libraries update this variable with the current position in the search. Use this value as the input parameter to subsequent calls.

- `pRequestList`

This is a pointer to the returned data. See the description of `DIVA_REQUEST_INFO` under the description of [DIVA_getRequestInfo](#).

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the *Manager*.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

DIVA_getFilesAndFolders

Retrieves the names of the files and folders for the specified object from DIVA. This function is included to support complex objects, but is valid for any object.

Set the `startIndex` to zero to get all of the file and folder names for an object. A list of names of the specified size is returned. Then set `startIndex` to the value of `nextStartIndex` and again make the function call. Continue this process until the return value equals `DIVA_WARN_NO_MORE_OBJECTS`.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getFilesAndFolders (
IN DIVA_STRING          objectName,
IN DIVA_STRING          objectCategory,
IN int                  listType,
IN int                  startIndex,
IN int                  batchSize,
IN DIVA_String          options,
OUT DIVA_FILES_AND_FOLDERS *pFilesAndFolders
);
```

- `objectName`

The name of the object to be queried.

- `objectCategory`

The Collection assigned to the object when it was archived.

- `listType`

Specifies what the returned list will include. See the definition of `DIVA_FILE_FOLDER_LIST_TYPE` later in this section.

- `startIndex`

The position in the list to start this iteration. Set at one (1) to start at the beginning. Values less than one are not valid. Set `startIndex` equal to `nextStartIndex` as returned in `DIVA_FILES_AND_FOLDERS` for all subsequent calls.

- batchSize

The maximum size of the returned list of objects. This must be set to a value no greater than 1000; the recommended setting is 500. This is only a suggestion and may be overridden by the underlying functionality. This parameter should not be used to guarantee that the list will be a certain size.

- options

Field for optional `getFilesAndFolders` terms.

- pFilesAndFolders

This is a pointer to the returned data. See the description of `DIVA_FILES_AND_FOLDERS` later in this section. It is the responsibility of the user to allocate and delete instances of this class.

```
typedef enum {
    DIVA_LIST_TYPE_FILES_ONLY = 0,
    DIVA_LIST_TYPE_FOLDERS_ONLY = 1,
    DIVA_LIST_TYPE_FILES_AND_FOLDERS = 2
} DIVA_FILE_FOLDER_LIST_TYPE;
```

- DIVA_LIST_TYPE_FILES_ONLY

This function will return files and symbolic links.

- DIVA_LIST_TYPE_FOLDERS_ONLY

This function will return folders only.

- DIVA_LIST_TYPE_FILES_AND_FOLDERS

This function will return files and folders and symbolic links.

```
class DIVA_FILES_AND_FOLDERS {
public:
    DIVA_OBJECT_SUMMARY          objectSummary;
    bool                          isComplex;
    int                          nextStartIndex;
    DIVA String                   siteName;
    vector<DIVA_FILE_FOLDER_INFO> *pFileFolderList;
};
```

- objectSummary

The ID of the object. See the description later in this section.

- isComplex

This is true when the object is a complex object.

- nextStartIndex

After the first and any subsequent call, the API libraries update this variable with the current position in the search. Use this value as the input parameter for subsequent calls.

- siteName

This contains the site name of the Manager that satisfied the job.

- `pFileFolderList`

This is a pointer to the list of files and folders. See the description of `DIVA_FILE_FOLDER_INFO` later in this section.

```
class DIVA_OBJECT_SUMMARY {
public:
    string      objectName;
    string      objectCategory;
};
```

- `objectName`

This is the name of the object.

- `objectCategory`

This is the Collection of the object.

```
class DIVA_FILE_FOLDER_INFO {
public:
    DIVA_STRING      fileOrFolderName;
    bool             isDirectory;
    bool             isSymbolicLink;
    __int64          sizeBytes;
    int              fileId;
    int              totalNumFilesFolders;
    __int64          totalSizeFilesFolders;
    vector<DIVA_CHECKSUM_INFO> pChecksumInfoList;
};
```

- `fileOrFolderName`

The name of the file or folder.

- `isDirectory`

This is true if the component is a directory.

- `isSymbolicLink`

This is true if the component is a symbolic link.

- `sizeBytes`

The size of the file in bytes. This is valid only for files.

- `fileId`

This is a unique ID for each file created by DIVA as part of the processing of this command.

- `totalNumFilesFolders`

The number of files and sub folders. This is valid only for folders in a complex object.

- `totalSizeFilesFolders`

The total size of all files, including files in sub folders. This is valid only for folders in a complex object.

- `pChecksumInfoList`

This is a pointer to a list of checksums for a file. Directories will not contain checksums. It is also possible that some files in the archive will not contain checksum information. See the description later in this section.

```
class DIVA_CHECKSUM_INFO {
public:
    DIVA_STRING      checksumType;
    DIVA_STRING      checksumValue;
    bool             isGenuine;
};
```

- `checksumType`

The type of checksum (MD5, SHA1, and so on).

- `checksumValue`

The value of the checksum in hexadecimal string format.

- `isGenuine`

This is true if this checksum was provided at the time of archiving and verified as a Genuine Checksum.

Return Values

The DIVA C++ API includes the following return values for this call:

- The file list contains empty files for non-complex objects.
- The folders list contains all folders in a non-complex object.
- Both the Folders Only and Files and Folders options are available for use with non-complex objects.

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_WARN_NO_MORE_OBJECTS`
The end of the list was reached during the call.

DIVA_getGroupsList

Returns the description of all Tape Groups. In DIVA 9.0 and later the Source Media Priority is reported in the returned data from this call.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getGroupsList (
OUT vector<DIVA_GROUP_DESC>      *&groups
);
```

- `groups`

This is a pointer to a list of `DIVA_GROUP_DESC` structures.

```
class DIVA_GROUP_DESC {
public:
string      group_name;
string      group_desc;
int         mediaFormatId;
};
```

- `group_name`

The configured name of the tape group.

- `group_desc`

The description of the tape group.

- `mediaFormatId`

The format of the tapes added to this Tape Group. The value can be `DIVA_MEDIA_FORMAT_LEGACY`, `DIVA_MEDIA_FORMAT_AXF`, or `DIVA_MEDIA_FORMAT_AXF_10`.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.

See also [DIVA_getObjectInfo](#).

DIVA_getObjectDetailsList

The `DIVA_getObjectDetailsList` is an API call to retrieve object information from the database. Only the latest state of the object is returned. Objects may be repeated across batches if the object is modified multiple times as the call advances (in time) from a user-specified time across objects in the database.

- The created-since call retrieves all objects created since a certain time.
- The deleted-since call retrieves all objects deleted since a certain time.
- If starting from a user-specified time of zero, the modified-since call retrieves all objects created since a certain time, and returns the state of the database from a time of zero.
- If starting from a user-specified time greater than zero, the call returns all objects created and deleted since a certain time, and all objects with newly created and (or) deleted instances.

In DIVA 9.0 and later storage options (at the instance level) are reported in the returned data from this call.

The `listPosition` vector returned by a `GetObjectDetailsList` call must be passed in to a subsequent call. Its content must not be altered by the user of the call.

Different detail levels can be specified (see the following Level of Detail Setting information). Level 0 will be the fastest, while Level 3 will return all possible details. Only the highest level of detail is supported. Using a lower level of detail will still return all information for objects.

The output can be structured using the `DIVA_OBJECTS_LIST` option, or through the `DIVA_TAPE_INFO_LIST` option. The output structure type is configured by setting the `pListType` parameter of the call.

The API client application should use the `DIVA_OBJECTS_LIST` setting in the following cases:

- To retrieve a list of objects instances added to DIVA.

- To retrieve a list of objects instances deleted from DIVA.
- To retrieve a combined list of all changes in the DIVA object database (adding and deleting objects, adding and deleting instances)
- To continuously monitor the DIVA system to retrieve events of adding and deleting objects, and adding and deleting instances.

The API client application should use the `DIVA_TAPE_INFO_LIST` setting to retrieve a list of tape instances for any instances added, deleted, repacked, ejected, or inserted.

Note: The `DIVA_TAPE_INFO_LIST` will not return any results for deleted instances if all objects are deleted.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getObjectDetailsList (
IN bool                                fFirstTime,
IN time_t                             *initialTime,
IN int                                pListType,
IN int                                pObjectsListType,
IN int                                pMaxListSize,
IN DIVA_STRING                        pObjectName,
IN DIVA_STRING                        pObjectCategory,
IN DIVA_STRING                        pMediaName,
DIVA_LEVEL_OF_DETAIL                  pLevelOfDetail,
IN vector<DIVA_STRING>                 listPosition,
OUT vector<DIVA_OBJECT_DETAILS_LIST>  *&pObjectDetailsList
);
```

- `fFirstTime`

The first time this function is called this parameter must be set to true. Every subsequent call should be set to false and `listPosition` must be copied from the `listPosition` value returned by the previous call to `DIVA_GetObjectDetailsList`.

- `initialTime`

The start time of the list. Data is collected and returned corresponding to this time and later. To retrieve all items in the database, use zero as the start time value.

- `pListType`

One of the codes defined by the enumeration `DIVA_LIST_TYPE`.

- `pObjectsListType`

One of the codes defined by the enumeration `DIVA_OBJECTS_LIST_TYPE`.

To retrieve all objects created, deleted, or modified since a certain time, set this to `DIVA_OBJECTS_CREATED_SINCE`, `DIVA_OBJECTS_DELETED_SINCE`, or `DIVA_OBJECTS_MODIFIED_SINCE`, respectively.

To retrieve tape related information for all objects that have been created, deleted, repacked, ejected, and (or) inserted since a certain time, set this parameter to

DIVA_INSTANCE_CREATED, DIVA_INSTANCE_DELETED, DIVA_INSTANCE_REPACKED, DIVA_INSTANCE_EJECTED, DIVA_INSTANCE_INSERTED, respectively.

To retrieve any combination of the above, use the pipe operator. For example, to retrieve tape information for objects with tape instances that have been created and repacked since a certain time, use `DIVA_INSTANCE_CREATED | DIVA_INSTANCE_REPACKED`.

- `pMaxListSize`

The maximum size of the returned list of objects. This must be set to a value no greater than 1000; the recommended setting is 500. This is only a suggestion and may be overridden by the underlying functionality. This parameter should not be used to guarantee that the list will be a certain size.

- `pObjectCategory`

Filter the returned list of objects based on the provided object Collection. The asterisk wildcard can be used (for example, `*video`).

- `pMediaName`

Filter the returned list of objects based on the provided media name. The asterisk wildcard can be used (for example, `soap*`).

- `pLevelOfDetail`

One of the codes defined by the enumeration `DIVA_LEVEL_OF_DETAIL`. Filtering by Object Name, Collection, and Tape Group (media name) is performed at all levels of detail.

The `DIVA_OBJECTS_CREATED_SINCE` and `DIVA_OBJECTS_MODIFIED_SINCE` options work with all levels of detail.

The `DIVA_OBJECTS_DELETED_SINCE` option only works with the `DIVA_OBJECTNAME_AND_CATEGORY` level of detail.

The `DIVA_TAPE_INFO_LIST` only works with the `DIVA_OBJECTNAME_AND_CATEGORY` and `DIVA_INSTANCE` level of detail.

- `listPosition`

A vector of `DIVA_STRING` type. The elements of this list are for internal use only and do not need to be extracted by the user.

When `pFirstTime` is true, a new empty list must be constructed and included.

When `pFirstTime` is false, `listPosition` must be updated with the `listPosition` attribute of `pObjectDetailsList` since this attribute points to the last object retrieved by the last call of `DIVA_getObjectDetailsList`.

- `pObjectDetailsList`

This is a pointer to the `DIVA_OBJECT_DETAILS_LIST` class. This is the output parameter that will contain the response to the call.

Use the `listPosition` parameter from this response as the `listPosition` argument in subsequent calls to `GetObjectDetailsList`.

For `pListType = DIVA_OBJECTS_LIST`, all of the object and (or) instance information is stored in the `objectInfo` attribute.

For `pListType = DIVA_TAPE_INFO_LIST`, all object and tape information is stored in the `objectTapeInfo` attribute.

```
typedef enum {
```

```
    DIVA_OBJECTNAME_AND_Category = 0,
    DIVA_MISC = 1,
    DIVA_COMPONENT = 2,
    DIVA_INSTANCE = 3
} DIVA_LEVEL_OF_DETAIL;
```

- `DIVA_OBJECTNAME_AND_CATEGORY (0)`

The `getObjectDetailsList` function will only return the object name and Collection.

- `DIVA_MISC (1)`

The `getObjectDetailsList` function will return the comments, archive date, name and path on the source, and all data returned with the `DIVA_OBJECTNAME_AND_CATEGORY` level of detail.

- `DIVA_COMPONENT (2)`

The `getObjectDetailsList` function will return the size of the object, list of components value, and all data returned with the `DIVA_MISC` level of details.

- `DIVA_INSTANCE (3)`

The `getObjectDetailsList` function will return all instance information, repack state, related active job information data, and all data returned with the `DIVA_COMPONENT` level of detail.

```
typedef enum {
```

```
    DIVA_OBJECTS_LIST = 1,
    DIVA_TAPE_INFO_LIST = 2
} DIVA_LIST_TYPE;
```

`DIVA_OBJECTS_LIST_TYPE` is defined as follows:

```
typedef enum {
```

```
    DIVA_OBJECTS_CREATED_SINCE = 0x0001,
    DIVA_OBJECTS_DELETED_SINCE = 0x0002,
    DIVA_OBJECTS_MODIFIED_SINCE = 0x0003,
    DIVA_INSTANCE_NONE = 0x0000,
    DIVA_INSTANCE_DELETED = 0x0020,
    DIVA_INSTANCE_REPACKED = 0x0040,
    DIVA_INSTANCE_EJECTED = 0x0080,
    DIVA_INSTANCE_INSERTED = 0x0100
```

```

} DIVA_OBJECTS_LIST_TYPE;

class DIVA_OBJECT_DETAILS_LIST {
public:
    int                                listType;
    DIVA_STRING                        siteID;
    vector<DIVA_STRING>                *listPosition;
    vector<DIVA_OBJECT_INFO>           *objectInfo;
    vector<DIVA_OBJECT_TAPE_INFO>      *objectTapeInfo;
};

```

- listType

One of the codes defined by the enumeration `DIVA_LIST_TYPE`.

- siteId

The DIVA system name as configured in `manager.conf`.

- listPosition

After the first and any subsequent call, the API libraries update this variable with the current position in the search. This object must be provided as the input parameter to any subsequent calls.

- objectInfo

This is a pointer to a `DIVA_OBJECT_INFO` structure. The structure should be allocated and deleted by the caller. The structure contains information about the object details, such as the list of components, tape instances, and other properties described in API call `getObjectInfo`.

- objectTapeInfo

This is a pointer to a list of `DIVA_OBJECT_TAPE_INFO` structures. The structure should be allocated and deleted by the caller. The structure contains information about the tapes containing instances of the object and other properties described in API call `getObjectTapeInfo`.

```

class DIVA_OBJECT_INFO {
public:
    DIVA_OBJECT_SUMMARY objectSummary;
    DIVA_STRING          uuid;
    int                  lockStatus;
    __int64              objectSize;
    __int64              objectSizeBytes;
    vector<string>        *filesList;
    string               objectComments;
    time_t               archivingDate;
    bool                 isInserted;
    vector<DIVA_TAPE_INSTANCE_DESC> *tapeInstances;
    vector<DIVA_ACTOR_INSTANCE_DESC> *actorInstances;
    string               objectSource;
    string               rootDirectory;
    vector<int>           *relatedRequests;
    bool                 toBeRepacked;
    int                  modifiedOrDeleted;
    bool                 isComplex;
    int                  nbFilesInComplexComponent;
    int                  nbFoldersInComplexComponent;
};

```

```
};
```

- `objectSummary`

The object name and Collection.

Note: Object Names cannot begin with a dollar sign (\$).

- `UUID`

Universally Unique Identifier to uniquely identify each object created in DIVA across all Telestream customer sites. This does not include objects created using Copy As jobs. An object created through a Copy As job will contain the same UUID as that of the Source Server object.

- `lockStatus`

This is the locking status of the object. Objects in the archive can be locked. When an object is locked it cannot be restored or copied to a new name. This feature prevents the use of an object that has an expired copyright, and so on. The object is unlocked when this value is zero.

- `objectSize`

This is the object size in kilobytes.

- `objectSizeBytes`

This is the object size in bytes.

- `filesList`

This is a list of the files in the object. A single wrapper file name is returned for complex objects.

- `objectComments`

This is the comments saved when the object was archived.

- `archivingDate`

Then number of seconds since January 1, 1970.

- `isInserted`

This is true if at least one instance of this object is either on a tape that is currently inserted in the Managed Storage, or a disk that is online.

- `tapeInstances`

This is a list of object instances saved to tape.

- `actorInstances`

This is a list of object instances saved to disk.

- `objectSource`

The Source Server system used to archive the object.

- `rootDirectory`

The root directory containing the object files on the objectsource.

- `relatedRequests`

This is non-terminated jobs.

- `toBeRepacked`

This is false unless all instances are going to be repacked.

- `modifiedOrDeleted`

One of `DIVA_MODIFIED_OR_DELETED` as follows:

- `UNDEFINED`—The `levelOfDetail` does not equal `DIVA_INSTANCE`.
- `DIVA_CREATED_OR_MODIFIED`—The object was created, or an instance was either added or removed.
- `DIVA_DELETED`—The object was removed.

- `isComplex`

This is true if this is a complex object.

- `nbFilesInComplexComponent`

This is the number of files in the object. This is used only for complex objects. The value is zero for non-complex objects.

- `nbFoldersInComplexComponent`

This is the number of folders in the object. This is used only for complex objects. The value is zero for non-complex objects.

```
class DIVA_OBJECT_SUMMARY {
public:
    string      objectName;
    string      objectCategory;
};
```

- `objectName`

This is the object name.

Note: Object Names cannot begin with a dollar sign (\$).

- `objectCategory`

This is the object Collection.

```
class DIVA_TAPE_INSTANCE_DESC {
public:
    int                instanceID;
    string             groupName;
    vector<DIVA_TAPE_DESC> *tapeDesc;
    bool               isInserted,
    DIVA_REQUIRE_STATUS reqStatus;
};
```

- `instanceId`

The numeric instance identifier.

- `groupName`

The name of the Tape Group this tape is assigned to.

- `tapeDesc`
Additional information about this tape.
- `isInserted`
This is true if at least one instance of this object is either on a tape that is currently inserted in the Managed Storage, or a disk that is online.
- `reqStatus`
Determines if the instance is Required or Released.
 - `DIVA_REQUIRED`—The instance is requested to be inserted into the Managed Storage.
 - `DIVA_RELEASED`—There is no need to have this instance present in the Managed Storage.

```
class DIVA_TAPE_DESC {
public:
    string          vsn;
    bool            isInserted;
    string          externalizationComment;
    bool            isGoingToBeRepacked;
    int             mediaFormatId;
};
```

- `vsn`
The volume serial number (barcode).
- `isInserted`
This is true if at least one instance of this object is either on a tape that is currently inserted in the Managed Storage or a disk that is online.
- `externalizedComment`
Comment saved when the tape was exported.
- `isGoingToBeRepacked`
This is false unless all instances are going to be repacked.
- `mediaFormatId`
The format of the data on to be used. The value can be `DIVA_MEDIA_FORMAT_DEFAULT`, `DIVA_MEDIA_FORMAT_LEGACY`, `DIVA_MEDIA_FORMAT_AXF`, or `DIVA_MEDIA_FORMAT_AXF_10`. This is only used when the `listType` is `Tape`.

```
typedef enum {
    DIVA_CLOUD_STORAGECLASS_NONE=0,
    DIVA_CLOUD_STORAGECLASS_ARCHIVE,
    DIVA_CLOUD_STORAGECLASS_STANDARD
} DIVA_CLOUD_STORAGECLASS;

class DIVA_ACTOR_INSTANCE_DESC {
public:
    int             instanceID;
    string          actor;
    DIVA_CLOUD_STORAGECLASS cloudStorageClass; (deprecated)
    DIVA_STRING     storageOptions;
};
```


- `instanceID`
The numeric ID of the instance.
- `actor`
This field reports the name of the disk array where the instance is stored instead of the Actor name.

```
typedef enum {
    DIVA_REQUIRED = 0,
    DIVA_RELEASED
} DIVA_REQUIRE_STATUS;

typedef enum {
    DIVA_UNDEFINED = 0,
    DIVA_CREATED_OR_MODIFIED,
    DIVA_DELETED
} DIVA_MODIFIED_OR_DELETED;
```

Return Values

The file list of each object in the objects list now contains empty files (that is, files of size 0 bytes). Client applications developed against DIVA C++ API releases before release 7.5 will receive empty files in the file list that accompanies a Details List message. Depending on the input terms, the `DIVA_getObjectDetailsList` function will return values as described in the following table.

List Type	Object List Type	Supported Detail Level	Return Value
DIVA_OBJECTS_LIST	DIVA_OBJECTS_CREATED_SINCE	All	List objects that have been created since a specified time.
DIVA_OBJECTS_LIST	DIVA_OBJECTS_DELETED_SINCE	Only DIVA_OBJECTNAME_AND_CATEGORY	List objects that have been deleted since a specified time.
DIVA_OBJECTS_LIST	DIVA_OBJECTS_MODIFIED_SINCE	Only DIVA_INSTANCE	List objects that have been created/deleted since a certain time, plus objects with new or deleted instances. If the list of instances is empty, objects were deleted. If the list of instances is not empty, objects were created or updated.

List Type	Object List Type	Supported Detail Level	Return Value
DIVA_TAPE_INFO_LIST	DIVA_INSTANCE_NONE (0x0000)	Only DIVA_OBJECTNAME_AND_CATEGORY and DIVA_INSTANCE level.	List objects and tape information for all tape instances (no filter).
DIVA_TAPE_INFO_LIST	DIVA_INSTANCE_CREATED (0x0010)	Only DIVA_OBJECTNAME_AND_CATEGORY and DIVA_INSTANCE level.	List objects and tape information for all tape instances created since a specified time.
DIVA_TAPE_INFO_LIST	DIVA_INSTANCE_DELETED (0x0020)	Only DIVA_OBJECTNAME_AND_CATEGORY and DIVA_INSTANCE level.	List objects and tape information for all tape instances deleted since a specified time.
DIVA_TAPE_INFO_LIST	DIVA_INSTANCE_REPACKED (0x0040)	Only DIVA_OBJECTNAME_AND_CATEGORY and DIVA_INSTANCE level.	List objects and tape information for all tape instances repacked since a specified time.
DIVA_TAPE_INFO_LIST	DIVA_INSTANCE_EJECTED (0x0080)	Only DIVA_OBJECTNAME_AND_CATEGORY and DIVA_INSTANCE level.	List objects and tape information for all tape instances ejected since a specified time.
DIVA_TAPE_INFO_LIST	DIVA_INSTANCE_INSERTED (0x0100)	Only DIVA_OBJECTNAME_AND_CATEGORY and DIVA_INSTANCE level.	List objects and tape information for all tape instances inserted since a specified time.

Use with DIVA Connect

All filters are applied at an object level as follows:

- If objects satisfying certain filter constraints are requested, those constraints are applied to the object and not to individual instances of an object.
- If an Object Name and Collection filter are specified, the list will be filtered to contain only objects satisfying the specified object name and collection.

Media name is defined at an instance level, not at an object level. A media name filter will only allow objects with at least one instance satisfying the job media name filter.

Note: If an instance of an object is created or deleted, and all modified objects with a particular media name are requested, the object will be returned if and only if any instance of the object satisfies the media name filter.

Example:

A new instance Object-A was added at time 101 with the media name CAR. Object-A has a total of two instances. One instance has the Media Name TRUCK and the other has the Media Name CAR.

An instance of Object-B was removed at time 101 with the Media Name CAR. Object-B has only one instance.

A new instance of Object-C was added at time 99 with the Media Name TRAIN. Object-C has a total of two instances. One instance has the Media Name TRAIN and the other has the Media Name HANG GLIDE.

A user executes a `getObjectDetailsList` call with `MODIFIED SINCE TIME 100` and `MEDIA NAME FILTER = T*`.

The only object that was modified since time 100, and has at least one instance with a Media Name of T is Object-A. Therefore, the result is that the list returned by the `getObjectDetailsList` call contains only Object-A.

Use and Recommended Practices

Telestream recommends that the DIVA C++ API client application adhere to the following sequence of actions:

1. Create a variable of `DIVA_OBJECT_DETAILS_LIST` type to store the object information returned by the call.
2. Create a variable of vector `<DIVA_STRING>` type to serve as the `listPosition` object. This will be used as the `listPosition` argument to `DIVA_GetObjectDetailsList`.
3. Create a variable of `time_t` type and set to the time at which the list is to start. Set this to zero to include all objects in the database.
4. Create a variable of Boolean type and set it to true to indicate that this is the first call in a sequence of calls.
5. Create a variables of Integer type to hold the `listType` and `objectsListType` to specify the type of call.
Example: Use `DIVA_OBJECTS_LIST` and `DIVA_OBJECTS_MODIFIED SINCE` to indicate that object information for modified objects is being requested.
6. Create a variable of Integer type to hold the suggested number of objects to be returned by the call.
7. Create list filtering variables of `DIVA_CHAR[]` type to hold the Object Name, Collection and Media filters.
8. Create a variable of Integer type to hold the level of detail to be returned.
9. Execute `DIVA_GetObjectDetailsList` with the variables previously mentioned.
10. Use the data stored in the variable from Step 1 as needed by the application.
11. Copy the `listPosition` attribute of the call's output created in Step 1 into the `listPosition` variable created in Step 2.
12. Repeat steps 8, 9, and 10 for until monitoring DIVA is no longer needed.
13. All variables must be deallocated after exiting the loop.

Multiple simultaneous calls to `DIVA_getObjectDetailsList` are supported. However, this call places a heavy demand on the database. Therefore simultaneous and (or) frequent calls to this function should be avoided.

Continuous monitoring of DIVA requires a procedure similar to the one defined in the section [Recommended Practices for Continuous Updates Notification Design Pattern \(No Media Filter\)](#).

Duplication of objects can occur across different return portions. It is important to handle these cases by examining the data returned by the call. For a `MODIFIED_SINCE` call, the instances of the duplicate object returned by successive calls must be compared to identify whether new information about the object is available and update the local repository accordingly.

An empty list may be returned as a valid result. This indicates that there were no changes to the system after the time specified in the last call. It is important to continue querying DIVA with the `DIVA_getObjectDetailsList` call using the ID from the previous call. However, the call frequency must be reduced after an empty list is received. This reduces the load on the database.

The same application can use the `DIVA_getObjectDetailsList` function effectively for both the initial database synchronization (if the client application maintains a database) and later use it for continuous monitoring after the database is updated.

During the initial database synchronization phase, it is necessary for the application to make frequent sequential calls to synchronize the local database with the database. The application must call `DIVA_getObjectDetailsList`, wait for a response, and then repeat the process.

After the synchronization phase, it is necessary for the application to go into the continuous monitoring phase, where it must make periodic calls to update the system with the latest object information. Telestream recommends a call interval of once every several minutes. Continuous, frequent execution of this call can heavily impact the database and degrade system performance.

The amount of data retrieved by the `CREATED_SINCE` and `MODIFIED_SINCE` call is substantial (object, instance, and component data for each object). Therefore, Telestream recommends that most applications use 500 as the maximum list size setting.

Recommended Practices for Continuous Updates Notification Design Pattern (No Media Filter)

The continuous updates notification design pattern is used in multiple applications, and is important when using the DIVA C++ API. The client application can use the internal database to continuously update the local database information with changes in the database. Following the design pattern helps develop the performance-optimized updates notification workflow.

The application must submit the call with the `objectListType` set to `MODIFIED_SINCE` with the level of detail required to collect instance-level information.

Additionally, the First Time flag must be set true, and all necessary filter terms must be set (object name and Collection).

This is the process the application will follow:

1. The application receives a list of objects and a new `listPosition`.
2. On the next cycle, the application will execute the call using the `listPosition` obtained in Step 1 and the First Time flag set to false. It is acceptable to submit another call immediately after receiving the list if the system is being used solely for synchronization purposes. Otherwise, it is recommended to wait for a period between calls to allow other jobs to process.
3. Repeat Steps 1 and 2 for the course of execution to keep the internal database synchronized with database.
4. If none of the objects in DIVA have been modified, the list will be EMPTY, which indicates there were no updates since the last call. The application should wait for a specific amount of time, and then retry.

The application must check the list of instances to see if the following occurred:

- The value of `modifiedOrDeleted` in the `DIVA_OBJECT_INFO` equals `DELETED`, objects were deleted and the database must be updated.
- The value of `modifiedOrDeleted` in the `DIVA_OBJECT_INFO` equals `CREATED_OR_MODIFIED`, the object was either created or updated.
 - If the object previously existed in the database, the database list of instances must be updated.
 - If the object does not exist in the database, it must be added to the database.

Note: To ensure continuous updates, the `listPosition` object should be preserved throughout the course of operations.

Example:

MAIN:

```
CREATE LIST_POSITION VARIABLE
CREATE DETAILS_LIST VARIABLE
SET FIRST_TIME = TRUE
SET INITIAL_TIME = 0
SET LIST_TYPE = DIVA_OBJECTS_LIST
SET OBJECTS_LIST_TYPE = DIVA_OBJECTS_MODIFIED_SINCE
SET LEVEL_OF_DETAIL = DIVA_OBJECTS_MODIFIED_SINCE
SET SIZE = 500
SET OBJECT_NAME = "*"
SET CATEGORY = "*"
SET MEDIA_NAME = "*"
CALL GetObjectDetailsList(FIRST_TIME, LIST_TYPE,
OBJECTS_LIST_TYPE, LIST_POSITION, SIZE, INITIAL_TIME, OBJECT_NAME,
CATEGORY, MEDIA_NAME, LEVEL_OF_DETAIL, DETAILS_LIST)
// 1

UNIQUE_ID AND DETAILS_LIST VARIABLES WERE UPDATED BY CALL // 2
```

```

CALL SYNC_OBJECTS                                     // 6

START LOOP
  SET FIRST TIME = FALSE
  CALL GetObjectDetailsList(...)                       // 3
  LIST_POSITION AND DETAILS_LIST VARIABLES WERE UPDATED BY CALL
  CALL SYNC_OBJECTS                                   // 6
END LOOP (TERMINATE AT END OF APPLICATION LIFE)        // 4

SYNC_OBJECTS:
  IF (DETAILS_LIST IS NOT EMPTY)                       // 5
    FOR (OBJECT IN DETAILS_LIST)
      IF (OBJECT.modifiedOrDeleted EQUALS DELETED)
        DELETE OBJECT FROM DATABASE                   // 6a
      ELSE
        IF (OBJECT.modifiedOrDeleted EQUALS CREATED_OR_MODIFIED)
          ADD OR UPDATE OBJECT TO DATABASE             // 6b
        END IF
      END IF
    END FOR
  END IF
END IF

```

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_WARN_NO_MORE_OBJECTS`
The end of the list was reached during the call.

DIVA_getObjectInfo

Returns information about a particular object in the DIVA system.

The `vector<DIVA_ACTOR_INSTANCE_DESC> *actorInstances` parameter is kept unchanged for compatibility, although it is formally a vector of `diskInstance` and not `actorInstance`.

The file list can contain empty files (that is, files of size 0 bytes). Client applications developed against DIVA C++ API releases before release 7.5 will also receive empty files in the file list that accompanies an `objectInfo` message.

For compatibility reasons, the class `DIVA_ACTOR_INSTANCE_DESC` designates a disk instance (not a Actor instance) and its string `actor` field now contains the array name instead of an Actor name.

In DIVA 9.0 and later, storage options (at the instance level) are reported in the returned data from this call.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getObjectInfo (
IN DIVA_STRING          objectName,
IN DIVA_STRING          objectCategory,
IN DIVA_STRING          options,
OUT DIVA_OBJECT_INFO    *objectInfo
);
```

- `objectName`

The name of the queried object.

- `objectCategory`

The Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.

- `options`

Optional string attribute for specifying additional terms to the job.

- `objectInfo`

Pointer to a `DIVA_OBJECT_INFO` structure allocated and deleted by the caller. See [DIVA_getObjectDetailsList](#) for a description of `DIVA_OBJECT_INFO`.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_OBJECT_DOESNT_EXIST`
The specified object does not exist in the DIVA Database.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the DIVA Database.

See also [DIVA_archiveObject](#), [DIVA_restoreObject](#), and [DIVA_deleteObject](#).

DIVA_getPartialRestoreRequestInfo

When processing the job `DIVA_PartialRestoreObject()`, and the format for the offsets were specified as timecodes, the offsets that are actually used may differ (somewhat) from what was specified in the job. After the Partial File Restore job is complete, this command can be used to obtain the actual offsets of the restored files.

This is a special purpose command that is valid only as follows:

- The job number to be queried must be a Partial File Restore job that has been successfully completed.
- The format specified in the Partial File Restore job must be a timecode type. This command is therefore not valid when the format of the job was folder-based or DPX.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getPartialRestoreRequestInfo (
IN int requestNumber,
OUT vector <DIVA_OFFSET_SOURCE_DEST> *fileList
);
```

- `requestNumber`

Identifies the completed Partial File Restore job to be queried.

- `fileList`

List of the files of an object that have been partially restored. Each structure contains the Source Server file name, a vector of the offsets used for the transfer, and a Destination Server file name. This vector must be similar to the vector provided to the `DIVA_partialRestoreObject()` function in terms of files and offset pairs. This function is provided to eventually detect that the actual offsets used for the transfer to the Destination Server have been adapted based on the format of the data to transfer.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_NO_SUCH_REQUEST`
The requestNumber identifies no job.
- `DIVA_ERR_INVALID_PARAMETER`
The requestNumber identifies no completed partial file restore job.

See also [DIVA_partialRestoreObject](#) and [DIVA_getRequestInfo](#).

DIVA_getRequestInfo

Obtains information about an archive, restore, delete, or repack job.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getRequestInfo (
IN int requestNumber,
OUT DIVA_REQUEST_INFO *requestInfo
);
```

- requestNumber

Identifies the queried job.

- requestInfo

Pointer to a `DIVA_REQUEST_INFO` structure. This is allocated and deleted by the caller.

```
class DIVA_REQUEST_INFO {
public:
int requestNumber;
DIVA_REQUEST_TYPE requestType;
DIVA_REQUEST_TYPE requestState;
DIVA_REQUEST_STATE requestState;
int progress;
DIVA_ABORTION_REASON abortionReason;
DIVA_OBJECT_SUMMARY objectSummary;
DIVA_REPACK_TAPES_INFO repackTapes;
int currentPriority;
DIVA_STRING additionalInfo;
time_t submissiondate;
time_t completiondate;
};
```

- requestNumber

The DIVA job number.

- requestType

See the definition of `DIVA_REQUEST_TYPE` later in this section.

- requestState

See the definition of `DIVA_REQUEST_STATE` later in this section.

- progress

The progress of the job from zero to one hundred percent if the `requestState` is `DIVA_TRANSFERRING` or `DIVA_MIGRATING`.

- abortionReason

The reason the job was terminated if the `requestState` is `DIVA_ABORTED`, otherwise this is zero.

- objectSummary

See the definition of `DIVA_OBJECT_SUMMARY` later in this section.

- repackTapes

Used if the `requestType` is `REPACK`.

- additionalInfo

See [Additional_Info](#) later in this section for use of this field.

- `submissionDate`

The date and time the job was submitted. This is UTC time in seconds (that is, seconds since January 1, 1970).

- `completionDate`

The date and time the job completed. This is UTC time in seconds and will be -1 if the job is still processing.

```
typedef enum {
    DIVA_ARCHIVE_REQUEST = 0,
    DIVA_RESTORE_REQUEST,
    DIVA_DELETE_REQUEST,
    DIVA_EJECT_REQUEST,
    DIVA_INSERT_REQUEST,
    DIVA_COPY_REQUEST,
    DIVA_COPY_TO_NEW_REQUEST,
    DIVA_RESTORE_INSTANCE_REQUEST,
    DIVA_DELETE_INSTANCE_REQUEST,
    DIVA_UNKNOW_REQUEST_TYPE,
    DIVA_AUTOMATIC_REPACK_REQUEST,
    DIVA_ONDEMAND_RAPACK_REQUEST,
    DIVA_ASSOC_COPY_REQUEST,
    DIVA_PARTIAL_RESTORE_REQUEST,
    DIVA_MULTIPLE_RESTORE_REQUEST,
    DIVA_TRANSCODE_ARCHIVED_REQUEST,
    DIVA_EXPORT_REQUEST,
    DIVA_TRANSFER_REQUEST,
    DIVA_AUTOMATIC_VERIFY_TAPES_REQUEST,
    DIVA_MANUAL_VERIFY_TAPES_REQUEST,
} DIVA_REQUEST_TYPE ;
```

```
typedef enum {
    DIVA_PENDING = 0,
    DIVA_TRANSFERRING,
    DIVA_MIGRATING,
    DIVA_COMPLETED,
    DIVA_ABORTED,
    DIVA_CANCELLED,
    DIVA_UNKNOWN_STATE,
    DIVA_DELETING,
    DIVA_WAITING_FOR_RESOURCES,
    DIVA_WAITING_FOR_OPERATOR,
    DIVA_ASSIGNING_POOL,
    DIVA_PARTIALLY_ABORTED,
    DIVA_RUNNING
} DIVA_REQUEST_STATE;
```

```
typedef enum {
    DIVA_AR_NONE = 0,
    DIVA_AR_DRIVE,
    DIVA_AR_TAPE,
    DIVA_AR_ACTOR,
    DIVA_AR_DISK,
    DIVA_AR_DISK_FULL,
    DIVA_AR_SOURCE_DEST,
    DIVA_AR_RESOURCES,
```

```
DIVA_AR_LIBRARY,  
DIVA_AR_PARAMETERS,  
DIVA_AR_UNKNOWN,  
DIVA_AR_INTERNAL,  
DIVA_AR_SOURCE_DEST2  
} DIVA_ABORTION_CODE;
```

- DIVA_AR_NONE = 0
Job not terminated.
- DIVA_AR_DRIVE
Drive trouble
- DIVA_AR_TAPE
Tape trouble
- DIVA_AR_ACTOR
Actor trouble
- DIVA_AR_DISK
Disk trouble
- DIVA_AR_DISK_FULL
The disk is full.
- DIVA_AR_SOURCE_DEST
Server trouble
- DIVA_AR_RESOURCES
Resource attribution trouble
- DIVA_AR_LIBRARY
Managed Storage trouble
- DIVA_AR_PARAMETERS
Incorrect Job terms
- DIVA_AR_UNKNOWN
Unknown code
- DIVA_AR_INTERNAL
Internal Manager error
- DIVA_AR_SOURCE_DEST2
This parameter has been deprecated but left intact for software compatibility.

```
class DIVA_ABORTION_REASON {  
public:  
    DIVA_ABORTION_CODE code;  
    string description;  
};
```

```
class DIVA_OBJECT_SUMMARY {  
public:  
    string      objectName;
```

```
string    objectCategory ;
};
```

- `objectName`
The name of the object.
- `objectCategory`
The Collection of the object.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The Job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_NO_SUCH_REQUEST`
The `requestNumber` identifies no job.
- `Additional_Info`
The `Additional_Info` field of the `DIVA_REQUEST_INFO` structure can contain one or more of the following depending on the job type:

MOB ID

MOB ID is a unique object identifier generated and used by AVID software. The DIVA C++ API provides the interface to retrieve the MOB ID for third party vendors after restoring archived objects to Unity. The MOB ID is available in the `additionalInfo` field of the `DIVA_REQUEST_INFO` structure. The MOB ID can be retrieved only when the object is restored to the AVID Unity system.

Example MOB ID:

060c2b34020511010104100013-000000-002e0815d552002b-060e2b347f7f-2a80

XML Document

Depending on the type of job, the XML document may be empty or it may contain any combination of the following elements. See the schema `additionalInfoRequestInfo.xsd` found in the `program\Common\schemas` folder of the DIVA installation.

When the job was a Restore, N-Restore, Partial File Restore, Copy, or Copy To New the list of media that contains the job's object is provided as follows:

```
<ADDITIONAL_INFO xmlns="http://www.telestream.net/diva/
additionalInfoRequestInfo/v1.0"> <Object>
  <Name>Object Name</Name>
  <Category>Collection</Category>
  <Instances>
    <DiskInstance>
      <Id>0</Id>
      <Disk>
        <MediaName>disk name</MediaName>
      </Disk>
    </DiskInstance>
    <TapeInstance>
      <Id>1</Id>
      <Tape>
        <MediaName>barcode</MediaName>
      </Tape>
    </TapeInstance>
  </Instances>
</Object>
</ADDITIONAL_INFO>
```

The following is included when the job was a Multiple Restore. If the restore is OK for one of the Destination Servers, but NOT OK for another, the Request State Parameter is `DIVA_PARTIALLY_ABORTED` and the Request Abortion Code is `DIVA_AR_SOURCE_DEST`. The status of each Destination Server is as follows:

```
<ADDITIONAL_INFO xmlns="http://www.telestream.net/diva/
additionalInfoRequestInfo/v1.0">
  <request id="12345" type="Restore">
    <destination name="destination name one" success="true"/>
    <destination name="destination name two" success="false"/>
  </request>
</ADDITIONAL_INFO>
```

The `ClipID` is included when the job was for a restore to a Quantel device. An ISA gateway never overwrites clips. A new `ClipID` is created for every imported clip. The `ClipID` of the created clip will be supplied after the Transfer Complete message as follows:

226 Transfer Complete. [new ClipID]

The Actor captures this new `ClipID` after the transfer and forwards it to the Manager. To use the API, `DIVA_GetRequestInfo` must be called. If the job is completed, the new `ClipID` will be in the Additional Request Information field as follows:

```
<ADDITIONAL_INFO xmlns="http://www.telestream.net/diva/
additionalInfoRequestInfo/v1.0">
  <ClipID>98765</ClipID>
</ADDITIONAL_INFO>
```

DIVA_getSourceDestinationList

This function returns a list of Source Servers present in a particular DIVA System.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS
DIVA_getSourceDestinationList (
    IN string                                options;
    OUT vector<DIVA_ACTOR_INSTANCE_DESC>    *&arraysInfo
)
```

- arraysInfo

Pointer to a list of DIVA_SOURCE_DESTINATION_LIST structures.

```
#ifndef WIN32
typedef long long __int64;
#endif

typedef enum {
    DIVA_SOURCE_TYPE_UNKNOWN = 0,
    DIVA_SOURCE_TYPE_MSS,
    DIVA_SOURCE_TYPE_PDR,
    DIVA_SOURCE_TYPE_SEACHANGE_BMC,
    DIVA_SOURCE_TYPE_SEACHANGE_BML,
    DIVA_SOURCE_TYPE_SEACHANGE_FTP,
    DIVA_SOURCE_TYPE_LEITCH,
    DIVA_SOURCE_TYPE_FTP_STANDARD,
    DIVA_SOURCE_TYPE_SFTP,
    DIVA_SOURCE_TYPE_DISK,
    DIVA_SOURCE_TYPE_LOCAL,
    DIVA_SOURCE_TYPE_CIFS,
    DIVA_SOURCE_TYPE_SIMULATION,
    DIVA_SOURCE_TYPE_OMNEON,
    DIVA_SOURCE_TYPE_MEDIAGRID,
    DIVA_SOURCE_TYPE_AVID_DHM,
    DIVA_SOURCE_TYPE_AVID_DET,
    DIVA_SOURCE_TYPE_AVID_AMC,
    DIVA_SOURCE_TYPE_QUANTEL_ISA,
    DIVA_SOURCE_TYPE_QUANTEL_QCP,
    DIVA_SOURCE_TYPE_SONY_HYPER_AGENT,
    DIVA_SOURCE_TYPE_METASOURCE,
    DATA_SOURCE_TYPE_MOVIETOME,
    DATA_SOURCE_TYPE_EXPEDAT,
    DATA_SOURCE_TYPE_AVID_DIRECT
} DIVA_SOURCE_TYPE;

class DIVA_SOURCE_DESTINATION_LIST{
public:
```

```

    DIVA_STRING server_Address;
    DIVA_STRING server_ConnectOption;
    int server_MaxAccess;
    int server_MaxReadAccess;
    __int64 server_MaxThroughput;
    int server_MaxWriteAccess;
    DIVA_STRING server_Name;
    DIVA_STRING server_ProductionSystem;
    DIVA_STRING server_RootPath;
    DIVA_SOURCE_TYPE server_SourceType;
};

```

- `server_Address`
The server IP address.
- `server_ConnectOption`
The server connection options.
- `server_MaxAccess`
The server maximum number of accesses.
- `server_MaxReadAccess`
The server maximum number of read accesses.
- `server_MaxThroughput`
The server maximum throughput.
- `server_MaxWriteAccess`
The server maximum write access.
- `server_Name`
The server name.
- `Server_ProductionSystem`
The server Network name.
- `server_RootPath`
The server root path.
- `server_SourceType`
The Source Server type.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the *Manager*.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

DIVA_getStoragePlanList

This function returns the list of Storage Plan Names that are defined in the DIVA system.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS          DIVA_getStoragePlanList (
IN string             options;
OUT vector<DIVA_STRING> *spList
);
```

- `spList`

A pointer to a list of Storage Plan Names.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

DIVA_getTapeInfo

Returns detailed information about a given tape identified by its barcode.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_getTapeInfo (
    IN DIVA_STRING barcode,
    OUT DIVA_DETAILED_TAPE_DESC *tapeInfo
);
```

- `barcode`

The barcode of the tape for which information is to be returned.

- `tapeInfo`

The returned information.

```
class DIVA_DETAILED_TAPE_DESC {
public:
    string      vsn;
    int         setID;
    string      group;
    int         typeID;
    string      type;
    int         fillingRatio;
    int         fragmentationRatio;
    __int64     remainingSize;
    __int64     totalSize;
    bool        isInserted;
    string      externalizationComment;
    bool        isGoingToBeRepacked;
    int         mediaFormatId;
};
```

- `setID`

Tape Set ID

- `typeID`

Tape Type ID

- `type`

Tape Type Name

- `fillingRatio`

The tape filling ratio using the equation:

`last_written_block / total_block_count`.

- `fragmentationRatio`

The tape fragmentation ration using the equation:

$$1 - (\text{valid_blocks_count}) / (\text{last_written_block})$$

Valid blocks are blocks used for archived objects not currently deleted.

- `mediaFormatId`

The format of the data on to be used. The value can be `DIVA_MEDIA_FORMAT_DEFAULT`, `DIVA_MEDIA_FORMAT_LEGACY`, `DIVA_MEDIA_FORMAT_AXF`, or `DIVA_MEDIA_FORMAT_AXF_10`.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

- `DIVA_ERR_TAPE_DOESNT_EXIST`

There is no tape associated with the given barcode.

DIVA_insertTape

Submits an Insert job to DIVA. This job completes when the operator has entered the job's tapes into the Managed Storage. The application is responsible for managing which tapes must be entered.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_insertTape (
IN bool      require,
IN int       priorityLevel,
OUT int      *requestNumber
)
```

```
DIVA_STATUS DIVA_insertTape (
IN bool      require,
IN int       priorityLevel,
IN int       acsId,
IN int       capId,
OUT int      *requestNumber
);
```

- `require`

When true, perform a `DIVA_require()` on every instance located on the successfully inserted tapes.

- `priorityLevel`

The priority level for this job. The `priorityLevel` can be in the range zero to one hundred, or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred, or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `acsId` (second form only)

The numeric ID of the ACS where the Insert operation must be executed.

When `acsId = -1` (default used for the first form), the Insert attempt will be performed in all known ACSs.

- `capId` (second form only)

The numeric ID of the CAP from where tapes will be inserted.

When `capId = -1` (default used for the first form), the Insert attempt will be performed in the first available CAP in the specified ACS.

- `requestNumber`

The number identifying the job.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is 300.

See also [DIVA_ejectTape](#).

DIVA_linkObjects

This function provides the opportunity to link together two existing objects; parent and child. If the objects are linked for Delete, anytime the parent object is deleted, the child will also be deleted. If objects are linked for Restore, anytime the parent object is restored, the child will be restored to the original location from where the child object was archived.

Synopsis

```
#include "DIVAapi.h"
```

```

DIVA_STATUS DIVA_linkObjects (
IN DIVA_STRING      parentName,
IN DIVA_STRING      parentCategory,
IN DIVA_STRING      childName,
IN DIVA_STRING      childCategory,

```

```
IN bool                cascadeDelete,
IN bool                cascadeRestore
);
```

- `parentName`
The parent object name.

Note: Object Names cannot begin with a dollar sign (\$).

- `parentCategory`
The parent object Collection.
- `childName`
The child object name.

Note: Object Names cannot begin with a dollar sign (\$).

- `childCategory`
The child object Collection.
- `cascadeDelete`
Indicates if the child object should be deleted along with parent.
- `cascadeRestore`
Indicates if the child object should be restored along with parent.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_OBJECT_ALREADY_EXISTS`
An object with this name and Collection already exists in the DIVA system.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.

DIVA_lockObject

A call to this function will lock an object. Locked objects cannot be restored.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_lockObject (
IN DIVA_STRING      objectName,
IN DIVA_STRING      Category,
IN string           options
);
```

- `objectName`
The name of the object.
- `Category`
The Collection assigned to the object when it was archived.
- `options`
Not currently in use.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system cannot accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.

DIVA_multipleRestoreObject

Submits an object Restore job to the Manager using several Destination Servers. The Manager chooses the appropriate instance to be restored. This function returns as soon as the Manager accepts the job.

The job will continue even if an error occurs with one of the Destination Servers. To check that the operation was successful the application must call the function `DIVA_getRequestInfo()`.

If `DIVA_MultipleRestoreObject()` is launched with a single Destination Server, the restore automatically converts to a `DIVA_RestoreObject()`.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_MultipleRestoreObject (
IN DIVA_STRING                      objectName,
IN DIVA_STRING                      objectCategory,
IN vector <DIVA_DESTINATION_INFO> destinations,
IN DIVA_RESTORE_QOS                qualityOfService,
IN int                             priorityLevel,
IN DIVA_STRING                      restoreOptions,
OUT int                             *requestNumber
)
public typedef struct _DIVA_DESTINATION_INFO {
DIVA_STRING                        destination;
DIVA_STRING                        filePathRoot;
} DIVA_DESTINATION_INFO,          *PDIVA_DESTINATION_INFO;
```

- `objectName`

The name of the object to be restored.

- `objectCategory`

The Collection assigned to the object when it was archived. This parameter can be a null string. However this may result in an error if several objects have the same name.

- `destinations`

A list of available Destination Servers (for example, a video server or browsing server) where object files can be restored. The names must be known by the DIVA configuration description.

A root folder where the object files will be placed is associated with each Destination Server. If null (`string("")`), the files will be placed in the `FILES_PATH_ROOT` folder specified when archiving the object using the `DIVA_archiveObject()` function.

- `qualityOfService`

One of the following codes:

- `DIVA_QOS_DEFAULT`

Restoring is performed according to the default Quality Of Service (currently direct and cache for restore operations).

- `DIVA_QOS_CACHE_ONLY`

Use cache restore only.

- `DIVA_QOS_DIRECT_ONLY`

Use direct restore only—no disk instance is created.

- `DIVA_QOS_CACHE_AND_DIRECT`

Use cache restore if available, or direct restore if cache restore is not available.

- `DIVA_QOS_DIRECT_AND_CACHE`

Use direct restore if available, or cache restore if direct restore is not available.

- `DIVA_QOS_NEARLINE_ONLY`

Use nearline restore only. Nearline restore will restore from a disk instance if a disk instance exists, otherwise, it will create a disk instance and restore from the newly created disk instance.

- `DIVA_QOS_NEARLINE_AND_DIRECT`

Use nearline restore if available, or direct restore if nearline restore 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_RESTORE_SERVICE_DO_NOT_OVERWRITE`

Do not overwrite existing files on the Destination Server.

- `priorityLevel`

The priority level for this job. The `priorityLevel` can be in the range zero to one hundred, or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred, or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `restoreOptions`

Additional options that must be used for performing the transfer of data from DIVA to the Destination Server. These options supersede any options specified in the DIVA configuration database. Currently the possible values for `restoreOptions` are:

- A null string to specify no objects
- `-login` represents the log in required for some Source Servers. This option obsoletes the `-gateway` option in earlier releases.
- `-pass` represents the password used with the `-login` option for some Source Servers.

- `requestNumber`

The job number assigned to this job. This number is used for querying the status or canceling the job.

Return Values

One of these `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

The DIVA system cannot accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`

An internal error was detected by the Manager or by the API.

- `DIVA_ERR_INVALID_PARAMETER`

A parameter value was not understood by the Manager.

- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`

The count of simultaneous jobs reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default is 300.

- `DIVA_ERR_OBJECT_DOESNT_EXIST`

The specified object does not exist in the database.

- `DIVA_ERR_OBJECT_OFFLINE`
There is no inserted instance in the Managed Storage and no Actor could provide a disk instance.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.
- `DIVA_ERR_OBJECT_IN_USE`
The object is currently in use (for example, Archived, Restored, Deleted, and so on).
- `DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST`
The specified Server is unknown by the DIVA system.
- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`
The specified object has instances that are partially deleted.

See also [DIVA_restoreObject](#), [DIVA_getRequestInfo](#), and [DIVA_copyToGroup and DIVA_copy](#).

DIVA_partialRestoreObject

Submits a Partial Object Restore job to the Manager and the Manager chooses the appropriate instance to be restored. This function returns as soon as the Manager accepts or rejects the job. To check that the operation was successful the application must call the `DIVA_getRequestInfo()` function.

If the job was not accepted (for example, if the job's object is on media not currently available) the job will generate an error. The media names (tape barcodes and disk names) that contain instances of the object are included in the `additionalInfo` field of the `DIVA_getRequestInfo()` response.

The Manager will use the `instanceID` field to select the instance of the object to use for the Partial Restore operation. The Manager will choose an appropriate instance to restore if `DIVA_ANY_INSTANCE` is used

DIVA supports four types of Partial Restore. The type implemented is determined by the format parameter in the job.

The following describes each type of Partial object Restore:

- `Byte Offset`
The format equals `DIVA_FORMAT_BYTES` and provides for a range of bytes to be extracted from a particular file in the archive. For example, bytes 1 to 2000 (the first 2000 bytes of the file) can be extracted, or byte 5000 to the end of the file (or both) and stored to an output file such as `movie.avi`.
The result of the Byte Offset Partial Restore is usually not playable when applied to video files. Actor will not apply the header, footer, and so on, according to the video format.
To issue a Byte Offset Partial Restore, pass `DIVA_FORMAT_BYTES` in the format field of the job. Create a `DIVA_OFFSET_SOURCE_DEST` object (in the `fileList` parameter

of the job). In the object the `sourceFile` in the archive and name the output file (`destFile`) must be specified. One or more `DIVA_OFFSET_PAIR` objects must be inserted within the `DIVA_OFFSET_SOURCE_DEST` object. These offset objects contain the ranges of bytes to be restored to the output file. The `fileFolder` and `range` fields within the `DIVA_OFFSET_SOURCE_DEST` object do not need to be populated.

Example:

```
start=10000 end=50000
```

- Timecode

The format equals `DIVA_FORMAT_VIDEO_*` and provides for a selected portion of a particular media file based on timecode. For example, 00:00:04:00 to 00:10:04:00 (a 10 minute segment starting 4 seconds in and ending at 10 minutes and 4 seconds) can be extracted and then place that segment into an output file such as `movie.avi`. The file is a smaller version of the original movie file.

The result of the Timecode Partial Restore is a valid clip when applied to video files. Actor will apply the header, footer, and so on, according to the video format. The job will be terminated if the Actor cannot parse the format. This type of Partial Restore can only be applied to a valid video clip.

To issue a Timecode Partial Restore, populate the `format` field in the job with the format of the file being partially restored. For example, if the file being restored is a GXF file, specify a value of `DIVA_FORMAT_VIDEO_GXF` in the `format` field of the job. DIVA provides an auto-detect feature that works for many types of media. Specify `DIVA_FORMAT_AUTODETECT` in the `format` field to use auto-detect.

Create a `DIVA_OFFSET_SOURCE_DEST` object in the `fileList` parameter of the job. In this object, add a `DIVA_OFFSET_PAIR` object using the `offsetVector` parameter that contains the start and end time. Use `DIVA_OFFSET_TC_END` to indicate the final timecode in the media file. The `fileFolder` and `range` fields within the `DIVA_OFFSET_SOURCE_DEST` object do not need to be populated.

Example:

```
start=01:01:01:00 end=02:02:02:00
```

- Files and Folders

Caution: In the following process The `offsetVector`, `sourceFile`, `destFile`, and `range` terms should not be specified for the Files and Folders Partial Object Restore type.

The format equals `DIVA_FORMAT_FOLDER_BASED` and provides for extracting entire files from the archive, or extracting entire directories and their contents. In DIVA multiple files and directories can be extracted in the same job. The files are restored with the file names and path names that were specified in the archive. No renaming option is valid in Files and Folders Partial Restore. For example, a file

archived as misc/12-2012/movie.avi would be partially restored to a misc/12-2012 subdirectory with the name movie.avi.

When a folder is specified in a Files and Folders Partial Restore, the folder and all files within that folder are restored. Each directory to be restored can have the `-r` option to recursively restore all folders nested within the target folder.

To issue a Files and Folders Partial Restore, the format field in the job must be populated with the `DIVA_FORMAT_FOLDER_BASED` value. Create a `DIVA_OFFSET_SOURCE_DEST` object in the `fileList` parameter of the job. In the object add a `DIVA_FILE_FOLDER` object in the `fileFolder` parameter containing the name of the file or folder to be restored, and any options (such as the recursive option) for that directory.

- DPX

The format equals `DIVA_FORMAT_DPX` and provides for extracting a range of DPX files from the archive. In this type of restore, the entire object is viewed as a single media item. One DPX file represents one frame of media. Only .dpx, .tif, and .tiff files in the archive are considered frames for the purposes of this command.

The first .dpx, .tif, or .tiff file in the archived object is considered Frame 1, the second .dpx in the archive is Frame 2, and so on.

For example, if frame 10 through frame 15 are extracted using DPX Partial Restore, it would restore the 10th .dpx file that appears in the archive, through (and including) the 15th .dpx file, resulting in six total files. Any other files (such as .wav files) are skipped by DPX Partial Restore.

Special frame numbers 0 and -1 may be used to refer to the first and last frame respectively. Frame 0 is valid as the start of a frame range and Frame -1 is valid as the end of a range.

Valid frames and ranges are as follows:

- Frame 0 = first frame
- Frame 1 = the first frame in the sequence.
- Frame n = the nth frame in the sequence.
- Frame -1 = last frame

Specifying frame 0 as the last frame is invalid.

Specifying Frame 0 to 0 is invalid and will not return the first frame as intended.

Specifying Frame 0 to 1 or Frame 1 to 1 will return the first frame.

Specifying the Frame -1 in the first frame produces an error. If the frame number of the last frame is unknown, Frame -1 to -1 cannot be specified to return the exact last frame.

Examples:

- `start=0 - end=1`

This will restore only the first frame.

- `start=600 - end=635, start=679 - end=779`

This will restore frames 600 through 635, and frames 679 through 779.

- `start=810 - end=-1`

This will restore all frames from frame 810 to the end of the archive.

Caution: In the following process the `offsetVector`, `sourceFile`, `destFile`, and `fileFolder` terms should not be specified for the DPX Partial Object Restore type.

To issue a DPX Partial Restore, populate the format field in the job with the value `DIVA_FORMAT_DPX`. Create a `DIVA_OFFSET_SOURCE_DEST` object in the `fileList` parameter of the job. In this object, add a `DIVA_RANGE` object in the `range` parameter that contains the start and end frames of the range to be restored.

To specify another range of frames within the same job, another `DIVA_OFFSET_SOURCE_DEST` object should be added to the job in the same manner.

The actual file name may, or may not, match the frame number in DIVA. During the restore process Manager interrogates the archive, finds the file order, and determines the frame number from the resulting file order. It does not consider the file name. The first .dpx, .tif, or .tiff file found is considered frame 1.

Be careful when archiving DPX files to ensure they can be partially restored properly, in part because DPX Partial Restore does not examine the file name or the DPX header information to determine which file is assigned to which frame. The assignment is based purely on the order in which the .dpx files appear in the archive. By default, the ordering is established by the Source Server and is typically alphanumeric. For example, NTFS DISK Servers order files and folders case insensitively as a general rule except where diacritical marks such as ' , ` ^, and so on are applied.

By default, when DIVA encounters a subfolder it recursively processes all of the children of that folder before continuing with other files. If a folder appears in the alphanumeric folder listing it is archived recursively in the order that it appears.

However, this can create some issues. For example, if all of the subdirectories of a given directory should be processed first, followed by the files in the directory, or all files processed first and then subdirectories is desired. The Actor allows the archive options `-file_order DIRS_FIRST` or `-file_order FILES_FIRST` to address these issues.

DPX Partial Restore looks at the entire object as a single piece of media. If multiple reels or clips appear in an archive they can be stored in folders and partially restored through a Files and Folders Partial Restore. However, they will be viewed as

one long movie clip to DPX Partial Restore. If this is desired, ensure that the directories are sorted alphanumerically in the order the frames should be arranged.

DIVA does not perform any special audio handling for DPX media other than what might be embedded in DPX files themselves. DIVA supports transcoding of DPX media. However, a transcoder may change the file names and (or) file order of the DPX archive.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_SPEC DIVA_partialRestoreObject (
IN string                                     objectName,
IN string                                     objectCategory,
IN int                                         instanceID,
IN vector <DIVA_OFFSET_SOURCE_DEST>          fileList,
IN string                                     destination,
IN string                                     filesPathRoot,
IN DIVA_RESTORE_QOS                          qualityOfService,
IN int                                         priorityLevel,
IN string                                     restoreOptions,
IN DIVA_FORMAT                               format,
OUT int                                        *requestNumber
);
```

- `objectName`

The name of the object to be partially restored.

- `objectCategory`

Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.

- `instanceID`

The ID of a non-spanned tape instance or `DIVA_ANY_INSTANCE`.

- `filelist`

List of the files of the object to be partially restored. Each structure contains the Source Server file name, a vector of offset pairs, and a Destination Server file name. The same source file can be used in several structures, but Destination Server files must be unique. A file present in the object cannot be in any structure or it won't be restored.

- `destination`

Destination Server (for example, a video server or browsing server) to put the object files. This name must be known by the DIVA configuration description.

- `filesPathRoot`

The root folder on the Destination Server where the object files will be placed. If this is null (`string("")`), the files will be placed in the `FILES_PATH_ROOT` folder specified when archiving the object using the `DIVA_archiveObject()` function.

- `qualityOfService`

One of the following codes:

- `DIVA_QOS_DEFAULT`

Restoring is performed according to the default Quality Of Service (currently direct restore).

- `DIVA_QOS_CACHE_ONLY (-qos_cache_only)`

Use cache restore only.

- `DIVA_QOS_DIRECT_ONLY (-qos_direct_only)`

Use direct restore only.

- `DIVA_QOS_CACHE_AND_DIRECT (-qos_cache_and_direct)`

Use cache restore if available, or direct restore if cache restore is not available.

- `DIVA_QOS_DIRECT_AND_CACHE (-qos_direct_and_cache)`

Use direct restore if available, or cache restore if direct restore 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_RESTORE_SERVICE_DO_NOT_OVERWRITE`

Do not overwrite existing files on the Destination Server.

- `priorityLevel`

The priority level for this job. The `priorityLevel` can be in the range zero to one hundred, or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the low-priority and one hundred the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred, or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `restoreOptions`

Additional options that must be used for performing the transfer of data from DIVA to the Destination Server. These options supersede any options specified in the DIVA configuration database. Currently the possible values for `restoreOptions` are as follows:

- A null string to specify no objects
- `-do_not_overwrite` executes this additional service

- `-do_not_check_existence` executes this additional service
- `-delete_and_write` executes this additional service
- `-login` represents the log in required for some Source Servers. This option obsoletes the `-gateway` option in earlier releases.
- `-pass` represents the password used with the `-login` option for some Source Servers.
- `format`
 - `DIVA_FORMAT_BYTES`
Offsets must be given as byte offsets. When the `offsetVector` field of a `DIVA_OFFSET_SOURCE_DEST` structure contains more than one `DIVA_OFFSET_PAIR` element, every corresponding extract is concatenated to create the Destination Server file.
 - `DIVA_FORMAT_BYTES_HEADER`
This has been deprecated but left for compatibility purposes only.
 - `DIVA_FORMAT_VIDEO_GXF`
Offsets must be given as timecodes, and the file to be partially restored must be in GXF format.
The `fileList` vector parameter must contain only one `DIVA_OFFSET_SOURCE_DEST` element.
The `offsetVector` vector parameter must contain only one `DIVA_OFFSET_PAIR` element.
Only the `DIVA_QOS_DIRECT_ONLY` Quality Of Service is supported for this format.
 - `DIVA_FORMAT_VIDEO_SEA`
Offsets must be given as timecodes. The file to be partially restored must be in SAF format and provide an index file.
A part description then contains one `DIVA_OFFSET_SOURCE_DEST` structure for each WAV file of the clip. There must be at least one WAV file per clip part.
 - * The Source Server file name in each structure must have the `.wav` or the `.WAV` extension.
 - * Each structure must contain exactly one `DIVA_OFFSET_PAIR` structure with a timecode pair equal to the timecode pair associated with the AVI file.
 - * The next part is delimited by the first `DIVA_OFFSET_SOURCE_DEST` structure associated with an AVI file.
 - * The Destination Server must support the successive restore of each part, with the AVI file (without WAV file) and then of the WAV files all at once in the same connection session.
 - `DIVA_FORMAT_VIDEO_MPEG2_TS`
Offsets must be given as timecodes. The video file must be encoded using the MPEG2 Transport Stream format. Use this for VELA encoders.

- `DIVA_FORMAT_VIDEO_MXF`

Offsets must be given as timecodes. The file format expected by this type of Partial File Restore is a single MXF file. A detailed matrix of supported MXF files is given in the product description.

- `DIVA_FORMAT_VIDEO_PINNACLE`

Offsets must be given as timecodes. This Partial File Restore format expects a specific object structure. This is applicable to Pinnacle clips composed of three files (header, ft, and std). DIVA prefers the MSS Server type for creating this clip.

The `fileList` vector parameter must contain only one `DIVA_OFFSET_SOURCE_DEST` element. The `offsetVector` must contain only one `DIVA_OFFSET_PAIR` element. The `DIVA_OFFSET_SOURCE_DEST` element must be associated with the header file only. The Destination Server name is also the header.

- `DIVA_FORMAT_VIDEO_OMNEON`

Offsets must be given as timecodes. This type of Partial File Restore can be used to partially restore QuickTime files (referenced and self-contained clips are supported). A detailed matrix of supported QuickTime clips is given in the product description.

The `fileList` vector parameter must contain only one `DIVA_OFFSET_SOURCE_DEST` element. The `offsetVector` must contain only one `DIVA_OFFSET_PAIR` element. The `DIVA_OFFSET_SOURCE_DEST` element must be associated with the .mov file only if it's not a self-contained clip.

- `DIVA_FORMAT_VIDEO_LEITCH`

Offsets must be given as timecodes. The video file must be encoded using the LEITCH Video Server and the format is LXF.

- `DIVA_FORMAT_VIDEO_QUANTEL`

Offsets must be given as timecodes. This type of Partial File Restore can be used to partially restore Quantel clips that have been archived with a `QUANTEL_QCP` Server type.

- `DIVA_FORMAT_AUTODETECT`

Offsets must be given as timecodes. This type of Partial File Restore can detect video clips with the following archive formats:

- * QuickTime self-contained
- * QuickTime with referenced media files (the .mov file must be in the first position)
- * DIF + WAV files
- * AVI with audio interleaved (separated WAV is not currently supported)
- * MXF (self-contained)
- * MPEG PS
- * LXF
- * Seachange (the .pd file must be in the first position)

The `fileList` vector parameter must contain only one `DIVA_OFFSET_SOURCE_DEST` element. The `offsetVector` must contain only one `DIVA_OFFSET_PAIR` element. The `DIVA_OFFSET_SOURCE_DEST` element must be associated with the following:

- * The .mov file if it is a QuickTime clip.
- * The .dif file if it is a DV file.
- * The .avi file if it is an AVI clip.

– `DIVA_FORMAT_FOLDER_BASED`

Specifies a set of files and folders to be restored. A recursive flag can be set to restore subfolders. All specified files and folders are restored.

– `DIVA_FORMAT_DPX`

Specifies a set of intervals, frame X through frame Y, where frames are sorted and traversed alphanumerically.

Only files with .tif or .tiff data formats are supported. All files must have a .dpx extension. The first frame of a DPX object is Frame 1. Frame numbers 0 and -1 can be used to refer to the first and last frame respectively.

- `requestNumber`

The job number assigned to this job. This number is used for querying the status or canceling this job.

```
class DIVA_OFFSET_SOURCE_DEST {
public:
    DIVA_STRING          sourceFile;
    vector<DIVA_OFFSET_PAIR> offsetVector;
    DIVA_STRING          destFile;
    DIVA_FILE_FOLDER     fileFolder;
    DIVA_RANGE           range;
};
```

- `sourceFile`

The Source Server file name when the format is other than `DIVA_FORMAT_FOLDER_BASED` or `DIVA_FORMAT_DPX`.

- `offsetVector`

The vector of intervals to restore. The type of all offsets in all `DIVA_OFFSET_SOURCE_DEST` structures must be compliant with the format parameter of the Partial File Restore job. Valid only when the format is other than `DIVA_FORMAT_FOLDER_BASED` or `DIVA_FORMAT_DPX`.

- `destFile`

The file name to be used at the Destination Server. Valid only when format is other than `DIVA_FORMAT_FOLDER_BASED` or `DIVA_FORMAT_DPX`.

- `fileFolder`

The file or folder name. Used only when the format is `DIVA_FORMAT_FOLDER_BASED`.

- range

The range of frames to be restored. Used only when the format is `DIVA_FORMAT_DPX`.

DIVA_OFFSET_PAIR (This class only has public functions.)

The following are the constructors:

- `DIVA_SPEC DIVA_OFFSET_PAIR (__int64 pBegin, __int64 pEnd, bool _isTimeCode)`

Constructor for use with byte offsets. `DIVA_OFFSET_BYTE_BEGIN` and `DIVA_OFFSET_BYTE_END` are valid.

- `DIVA_SPEC DIVA_OFFSET_PAIR (const DIVA_STRING &pBegin, const DIVA_STRING &pEnd)`

Constructor for use with timecode offsets. Timecodes are formatted as HH:MM:SS:FF.

The following are the attribute accessors:

- `DIVA_SPEC bool isTimeCode();`

This is true if the offset pair was constructed with timecode offsets.

- `DIVA_SPEC DIVA_STRING getTimeCodeBegin();`

Return the beginning offset as a timecode.

- `DIVA_SPEC DIVA_STRING getTimeCodeEnd();`

Return the ending offset as a timecode.

- `DIVA_SPEC __int64 getByteBegin();`

Return the beginning offset as bytes.

- `DIVA_SPEC __int64 getByteEnd();`

Return the ending offset as bytes.

```
class DIVA_FILE_FOLDER {
public:
    DIVA_STRING    fileFolder;
    DIVA_STRING    option
};
```

- fileFolder

The file or folder name.

- option

Options (for example, -r to recurse folders).

```
class DIVA_RANGE {
public:
    int    startRange;
    int    endRange;
};
```

- `startRange`
The first frame number to be restored.
- `endRange`
The last frame number to be restored.

The format gives information about how to interpret the interval and about which specific operation should eventually be performed.

```
typedef enum {
    DIVA_FORMAT_BYTES = 0,
    DIVA_FORMAT_BYTES_HEADER,
    DIVA_FORMAT_VIDEO_GXF,
    DIVA_FORMAT_VIDEO_SEA,
    DIVA_FORMAT_VIDEO_AVI_MATROX,
    DIVA_FORMAT_VIDEO_MPEG2_TS,
    DIVA_FORMAT_VIDEO_MXF,
    DIVA_FORMAT_VIDEO_PINNACLE,
    DIVA_FORMAT_VIDEO_OMNEON,
    DIVA_FORMAT_VIDEO_LEITCH,
    DIVA_FORMAT_VIDEO_QUANTEL,
    DIVA_FORMAT_AUTODETECT,
    DIVA_FORMAT_FOLDER_BASED,
    DIVA_FORMAT_DPX
} DIVA_FORMAT;
```

- `DIVA_FORMAT_BYTES`
Raw bytes
- `DIVA_FORMAT_VIDEO_GXF`
GXF video format
- `DIVA_FORMAT_VIDEO_SEA`
Seachange video format
- `DIVA_FORMAT_VIDEO_AVI_MATROX`
Matrox-specific AVI format (+ WAV files)
- `DIVA_FORMAT_VIDEO_MPEG2_TS`
MPEG Transport Stream
- `DIVA_FORMAT_VIDEO_MXF`
MXF video format
- `DIVA_FORMAT_VIDEO_PINNACLE`
Pinnacle video format
- `DIVA_FORMAT_VIDEO_OMNEON`
Omneon video format
- `DIVA_FORMAT_VIDEO_LEITCH`
Leitch video format
- `DIVA_FORMAT_VIDEO_QUANTEL`
Quantel QCP video format

- `DIVA_FORMAT_VIDEO_AUTODETECT`
Automatic format detection
- `DIVA_FORMAT_FOLDER_BASED`
Fully restore the specified files and (or) folders
- `DIVA_FORMAT_DPX`
DPX video format

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
DIVA can no longer accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set using the `DIVA_API_TIMEOUT` variable. The default value is one hundred-eighty (180) seconds.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
The Manager or API detected an internal error.
- `DIVA_ERR_INVALID_PARAMETER`
The Manager did not understand a parameter value.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is three hundred.
- `DIVA_ERR_OBJECT_DOESNT_EXIST`
The specified object does not exist in the database.
- `DIVA_ERR_OBJECT_OFFLINE`
There is no inserted instance in the Managed Storage and no Actor could provide a disk instance.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.

- `DIVA_ERR_INSTANCE_OFFLINE`
The instance specified for restoring this object is ejected, or the Actor owning the specified disk instance is not available.
- `DIVA_ERR_INSTANCE_DOESNT_EXIST`
The instance specified for restoring this object does not exist.
- `DIVA_ERR_OBJECT_IN_USE`
The object is currently in use (being Archived, Restored, Deleted, and so on).
- `DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST`
The specified Server is unknown by the DIVA system.
- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`
The specified object has instances that are partially deleted.

See also [DIVA_restoreObject](#), [DIVA_getRequestInfo](#), and [DIVA_getPartialRestoreRequestInfo](#).

DIVA_release

Indicates to the Manager that this instance can be externalized. This function has no effect if the instance has already been released. The list of instances that are `RELEASED` and `INSERTED` may be retrieved and shown in the Web App.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_release (
    IN DIVA_STRING      objectName,
    IN DIVA_STRING      categoryName,
    IN int               instanceID
);
```

- `objectName`
The name of the object to be copied.
- `objectCategory`
The Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.
- `instanceID`
A value of `DIVA_EVERY_INSTANCE` forces this function to apply to every instance of the given object.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
DIVA can no longer accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set using the `DIVA_API_TIMEOUT` variable. The default value is one hundred-eighty (180) seconds.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the *Manager*.
- `DIVA_ERR_INTERNAL`
The Manager or API detected an internal error.
- `DIVA_ERR_INVALID_PARAMETER`
The Manager did not understand a parameter value.
- `DIVA_ERR_OBJECT_DOESNT_EXIST`
The specified object does not exist in the database.
- `DIVA_ERR_INSTANCE_DOESNT_EXIST`
The instance specified for restoring this object does not exist.
- `DIVA_ERR_INSTANCE_MUST_BE_ON_TAPE`
No tape instance exists for this object.
- `DIVA_ERR_NO_INSTANCE_TAPE_EXIST`
The specified object has instances that are partially deleted.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.

See also [DIVA_require](#).

DIVA_require

Indicates to the Manager that this instance must be inserted. If the instance is already inserted, this function has no effect. The list of instances that are `REQUIRED` and `EJECTED` can be retrieved and shown in the Web App.

Synopsis

```
#include "DIVAapi.h"
```



```
DIVA_STATUS DIVA_require(
    IN DIVA_STRING      objectName,
    IN DIVA_STRING      categoryName,
    IN int               instanceID
);
```

- `objectName`
Name of the object to be copied.
- `objectCategory`
Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.
- `instanceID`
A value of `DIVA_EVERY_INSTANCE` forces the function to apply to every instance of the given object.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
DIVA can no longer accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set using the `DIVA_API_TIMEOUT` variable. The default value is one hundred-eighty (180) seconds.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the *Manager*.
- `DIVA_ERR_INTERNAL`
The Manager or API detected an internal error.
- `DIVA_ERR_INVALID_PARAMETER`
The Manager did not understand a parameter value.
- `DIVA_ERR_OBJECT_DOESNT_EXIST`
The specified object does not exist in the database.

- `DIVA_ERR_INSTANCE_DOESNT_EXIST`
The instance specified for restoring this object does not exist.
- `DIVA_ERR_INSTANCE_MUST_BE_ON_TAPE`
No tape instance exists for this object.
- `DIVA_ERR_NO_INSTANCE_TAPE_EXIST`
The specified object has instances that are partially deleted.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.

See also [DIVA_release](#).

DIVA_restoreInstance

Restores an object from a specific instance. If the instance is externalized the operation fails even if there are other instances available for the object.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_restoreInstance (
IN DIVA_STRING          objectName,
IN DIVA_STRING          CategoryName,
IN int                  instanceID,
IN DIVA_STRING          destination,
IN DIVA_STRING          filePathRoot,
IN DIVA_RESTORE_QOS     qualityOfService,
IN int                  priorityLevel,
IN DIVA_STRING          restoreOptions,
OUT int                 *requestNumber
);
```

- `objectName`
Name of the object to be restored.
- `objectCategory`
Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.
- `instanceID`
The instance identifier.
- `destination`
The Destination Server (for example, a video server or browsing server) where the object files will be restored. This name must be known by the DIVA configuration description.

- `filePathRoot`

Root folder on the Destination Server where the object files will be placed. If this is null (`string("")`), the files will be placed in the `FILES_PATH_ROOT` folder specified when archiving the object using the `DIVA_archiveObject()` function.

- `qualityOfService`

One of the following codes:

- `DIVA_QOS_DEFAULT`

Restoring is performed according to the default Quality Of Service (currently direct and cache for restore operations).

- `DIVA_QOS_CACHE_ONLY`

Use cache archive only.

- `DIVA_QOS_DIRECT_ONLY`

Use direct restore only—no disk instance is created.

- `DIVA_QOS_CACHE_AND_DIRECT`

Use cache restore if available, or direct restore if cache restore is not available.

- `DIVA_QOS_DIRECT_AND_CACHE`

Use direct restore if available, or cache restore if direct restore is not available.

Additional and optional services are available. To job those services, use a logical OR between the previously documented Quality Of Service parameter and the following constant:

* `DIVA_RESTORE_SERVICE_DO_NOT_OVERWRITE`

Do not overwrite existing files on the Destination Server.

- `priorityLevel`

The priority level for this job. The `priorityLevel` can be in the range zero to one hundred, or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred, or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `restoreOptions`

Additional options that must be used for performing the transfer of data from DIVA to the Destination Server. These options supersede any options specified in the

DIVA configuration database. Currently the possible values for `restoreOptions` are as follows:

- Null String

A null string specifies no options.

- `-login`

A user name and password is required to log in to some Source Servers. This option obsoletes the `-gateway` option from earlier releases.

- `-pass`

The password used with `-login`.

- `requestNumber`

A number identifying this job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`

No connection is open.

- `DIVA_ERR_SYSTEM_IDLE`

DIVA can no longer accept connections and queries.

- `DIVA_ERR_BROKEN_CONNECTION`

The connection with the Manager was broken.

- `DIVA_ERR_TIMEOUT`

The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set using the `DIVA_API_TIMEOUT` variable. The default value is one hundred-eighty (180) seconds.

- `DIVA_ERR_UNKNOWN`

An unknown status was received from the Manager.

- `DIVA_ERR_INTERNAL`

The Manager or API detected an internal error.

- `DIVA_ERR_INVALID_PARAMETER`

The Manager did not understand a parameter value.

- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`

Count of simultaneous jobs has reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default is 300.

- `DIVA_ERR_OBJECT_DOESNT_EXIST`

The specified object does not exist in the database.

- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.
- `DIVA_ERR_INSTANCE_OFFLINE`
The specified instance for restoring this object is ejected, or the Actor owning the specified disk instance is not available.
- `DIVA_ERR_INSTANCE_DOESNT_EXIST`
The instance specified for restoring this object does not exist.
- `DIVA_ERR_OBJECT_IN_USE`
The object is currently in use (being Archived, Restored, Deleted, and so on).
- `DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST`
The specified Server is not known by the DIVA system.
- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`
The specified object has instances that are partially deleted.

See also [DIVA_archiveObject](#) and [DIVA_getObjectInfo](#).

DIVA_restoreObject

Submits an Object Restore job to the Manager and the Manager chooses the appropriate instance to be restored. This function returns as soon as the Manager accepts the job. To check that the operation was successful, the application must call the function `DIVA_getRequestInfo()`.

If the job's object is on media that is not available, the job will fail. The media names (tape barcodes and disk names) that contain instances of the object will be included in the `additionalInfo` field of the `DIVA_getRequestInfo()` response.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_restoreObject (
IN  DIVA_STRING      objectName,
IN  DIVA_STRING      objectCategory,
IN  DIVA_STRING      destination,
IN  DIVA_STRING      filePathRoot,
IN  DIVA_RESTORE_QOS qualityOfService,
IN  int              priorityLevel,
IN  DIVA_STRING      restoreOptions,
OUT int              *requestNumber
);
```

- `objectName`
Name of the object to be restored.

- `objectCategory`

Collection assigned to the object when it was archived. This parameter can be a null string. However, this may result in an error if several objects have the same name.

- `destination`

The Destination Server (for example, a video server or browsing server) where the object files will be restored. This name must be known by the DIVA configuration description.

- `filePathRoot`

Root folder on the Destination Server where the object files will be placed. If this is null (`string("")`), the files will be placed in the `FILES_PATH_ROOT` folder specified when archiving the object using the `DIVA_archiveObject()` function.

- `qualityOfService`

One of the following codes:

- `DIVA_QOS_DEFAULT`

Restoring is performed according to the default Quality Of Service (currently direct and cache for restore operations).

- `DIVA_QOS_CACHE_ONLY(-qos_cache_only)`

Use cache restore only.

- `DIVA_QOS_DIRECT_ONLY(-qos_direct_only)`

Use direct restore only.

- `DIVA_QOS_CACHE_AND_DIRECT(-qos_cache_and_direct)`

Use cache restore if available, or direct restore if cache restore is not available.

- `DIVA_QOS_DIRECT_AND_CACHE(-qos_direct_and_cache)`

Use direct restore if available, or cache restore if direct restore 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_QOS_NEARLINE_ONLY(-qos_nearline_only)`

Use Nearline Restore only. Nearline Restore will restore from a disk instance if it exists, otherwise, it will create a disk instance and restore from the newly created disk instance.

- `DIVA_QOS_NEARLINE_AND_DIRECT(-qos_nearline_and_direct)`

Use Nearline Restore if available, or Direct Restore if Nearline Restore 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 constants:

* `DIVA_RESTORE_SERVICE_DO_NOT_OVERWRITE`

Do not overwrite existing files on the Destination Server.

* `DIVA_RESTORE_SERVICE_DO_NOT_CHECK_EXISTENCE`

Do not check existence of the clip on the server.

* `DIVA_RESTORE_SERVICE_DELETE_AND_WRITE`

Force delete and rewrite if object exists on the server.

* `DIVA_RESTORE_SERVICE_DEFAULT`

Operate using the default setting in the Manager configuration.

- `priorityLevel`

The priority level for this job. The `priorityLevel` can be in the range zero to one hundred, or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred, or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `restoreOptions`

Additional options that must be used for performing the transfer of data from DIVA to the Destination Server. These options supersede any options specified in the DIVA configuration database. Currently the possible values for `restoreOptions` are as follows:

- Null String

A null string specifies no options.

- `-login`

A user name and password is required to log in to some Source Servers. This option obsoletes the `-gateway` option from earlier releases.

- `-pass`

The password used with `-login`.

- `requestNumber`

Job number assigned to this job. This number is used for querying the status or canceling this job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
DIVA can no longer accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set using the `DIVA_API_TIMEOUT` variable. The default value is one hundred-eighty (180) seconds.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
The Manager or API detected an internal error.
- `DIVA_ERR_INVALID_PARAMETER`
The Manager did not understand a parameter value.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is three hundred.
- `DIVA_ERR_OBJECT_DOESNT_EXIST`
The specified object does not exist in the database.
- `DIVA_ERR_OBJECT_OFFLINE`
There is no inserted instance in the Managed Storage and no Actor could provide a Disk Instance.
- `DIVA_ERR_SEVERAL_OBJECTS`
More than one object with the specified name exists in the database.
- `DIVA_ERR_OBJECT_IN_USE`
The object is currently in use (being Archived, Restored, Deleted, and so on).
- `DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST`
The specified Server is not known by the DIVA system.
- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`
The specified object has instances that are partially deleted.

See also [DIVA_getRequestInfo](#) and [DIVA_copyToGroup and DIVA_copy](#).

DIVA_transcodeArchive

Submits a Transcode Archive job to the Manager. The original object will be restored to the local Actor cache then transcoded to the format defined in the option field. A new object containing the transcoded clip will then be archived back to DIVA.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_transcodeArchive (
IN DIVA_STRING          parentObjectName,
IN DIVA_STRING          parentObjectCategory,
IN int                  instance,
IN DIVA_STRING          objectName,
IN DIVA_STRING          objectCategory,
IN DIVA_STRING          mediaName,
IN DIVA_STRING          comments,
IN DIVA_STRING          archiveOptions,
IN DIVA_ARCHIVE_QOS     qualityOfService,
IN bool                 bCascadeDelete,
IN int                  priorityLevel,
OUT int                 *requestNumber
);
```

- `parentObjectName`
Name of the original object to be transcoded.
- `parentObjectCategory`
Collection assigned to the original object.
- `instance`
Instance of the parent object. The default is -1.
- `objectName`
Name of the resulting transcoded object from the transcoding operation.
- `objectCategory`
Collection of the transcoded object.

- `mediaName`

The tape group or disk array where the object is to be saved. The media may be defined as follows:

- `Name` (of the Tape Group or Array)

Provide the Tape Group or Disk Array name as defined in the configuration. The object is saved to the specified media and assigned to the default SP (Storage Plan).

- `SP Name`

Provide a Storage Plan Name as defined in the configuration. The object will be assigned to the specified Storage Plan and saved to the default media specified.

- Both of the above (`Name` and `SP Name`)

The object is saved to the specified media as in `Name`, and assigned to the specified Storage Plan as in `SP Name`. The `Name` and the `SP Name` must be separated by the `&` delimiter (this is configurable).

When this parameter is a null string, the default group of tapes called `DEFAULT` is used. Complex objects can only be saved to AXF media types.

- `comments`

Optional information describing the object. This can be a null string.

- `archiveOptions`

Additional options that must be used for performing the transfer of data from the Source Server to DIVA. These options supersede any options specified in the DIVA configuration database. Currently the possible values for `archiveOptions` are:

- `-tr_archive_format FORMAT`

Destination Server format of the retrieved object. This is required.

- `-tr_names trans1`

Names of the transcoders that have to perform this operation. If more than one transcoder is selected, the performing transcoder will be chosen based on the current loading. If this option is not specified, the performing transcoder will be chosen from all DIVA transcoders based on the current loading. This is optional.

- `-tr_names trans1,trans2`

Names of the transcoders that have to perform this operation. Multiple transcoders are identified in a comma separated list (`trans1`, `trans2`, and so on). If more than one transcoder is selected, the performing transcoder will be chosen based on the current loading. If this option is not specified, the performing transcoder will be chosen from all DIVA transcoders based on the current loading. This is optional.

- `qualityOfService`

One of the following codes:

- `DIVA_QOS_DEFAULT`

Restoring is performed according to the default Quality Of Service (currently cache for archive operations).

- `DIVA_QOS_CACHE_ONLY`

Use cache archive only.

- `DIVA_QOS_DIRECT_ONLY`

Use direct archive only—no disk instance is created.

- `DIVA_QOS_CACHE_AND_DIRECT`

Use cache archive if available, or direct archive if cache archive is not available.

- `DIVA_QOS_DIRECT_AND_CACHE`

Use direct archive if available, or cache archive if direct archive is not available.

- `bCascadeDelete`

Shows if transcoded object is linked to the original object. If true, both the original object and the transcoded object will be deleted.

- `priorityLevel`

The priority level for this job. The `priorityLevel` can be in the range zero to one hundred, or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred the highest priority.

- There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred, or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `requestNumber`

Job number assigned to this job. This number is used for querying the status or canceling this job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`

The job was correctly submitted and accepted by the Manager.

- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system can no longer accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set using the `DIVA_API_TIMEOUT` variable. The default value is one hundred-eighty (180) seconds.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs reached the maximum allowed value. This variable is set in the `manager.conf` configuration file. The default value is three hundred.
- `DIVA_ERR_OBJECT_ALREADY_EXISTS`
The specified object already exists in the database.
- `DIVA_ERR_OBJECT_PARTIALLY_DELETED`
The specified object has instances that are partially deleted.

See also [DIVA_linkObjects](#).

DIVA_transferFiles

Submits a Transfer Files job to the Manager. The job will transfer files from a remote server (the Source Server) to another remote server (the Destination Server). This function returns as soon as the Manager accepts the job. The application must call the function `DIVA_getRequestInfo()` to confirm that the operation completed successfully.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_transferFiles (
IN DIVA_STRING                source,
IN DIVA_STRING                sourcePathRoot,
IN vector<DIVA_STRING>        filenamesList,
IN DIVA_STRING                destination,
```

```
IN DIVA_STRING      destinationPathRoot,
IN int              priorityLevel,
OUT int             *requestNumber
);
```

- `source`

Name of the Source Server (for example, a video server or browsing server). This name must be known by the DIVA configuration description.

- `sourcePathRoot`

Root folder for the files specified by the `filenamesList` parameter.

- `filenamesList`

List of file path names relative to the folder specified by the `sourcePathRoot` parameter. When the `sourcePathRoot` is null, path names must be absolute names.

- `destination`

Name of the Destination Server (for example a video server or browsing server). This name must be known by the DIVA configuration description.

- `destinationPathRoot`

Root folder where the files will be placed at the Destination Server.

- `priorityLevel`

The priority level for this job. The `priorityLevel` can be in the range zero to one hundred, or the value `DIVA_DEFAULT_REQUEST_PRIORITY`. The value zero is the lowest priority and one hundred the highest priority.

There are six predefined values as follows:

- `DIVA_REQUEST_PRIORITY_MIN`
- `DIVA_REQUEST_PRIORITY_LOW`
- `DIVA_REQUEST_PRIORITY_NORMAL`
- `DIVA_REQUEST_PRIORITY_HIGH`
- `DIVA_REQUEST_PRIORITY_MAX`
- `DIVA_DEFAULT_REQUEST_PRIORITY`

When the `DIVA_DEFAULT_REQUEST_PRIORITY` value is used, the Manager uses the default priority defined in the Manager configuration for the job.

Using a value either outside of the range of zero to one hundred, or predefined values yields a `DIVA_ERR_INVALID_PARAMETER` error.

- `requestNumber`

Job number assigned to this job. This number is used for querying the status or canceling this job.

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system is no longer able to accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.
- `DIVA_ERR_INVALID_PARAMETER`
A parameter value was not understood by the Manager.
- `DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`
The count of simultaneous jobs reached the maximum allowed value. This variable is set in the `manager.conf` configuration file and the default value is three hundred.
- `DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST`
The specified Server is not known by the DIVA system.

Also see [DIVA_getRequestInfo](#).

DIVA_unlockObject

A call to this function will unlock an object. Locked objects cannot be restored.

Synopsis

```
#include "DIVAapi.h"
```

```
DIVA_STATUS DIVA_unlockObject (
IN DIVA_STRING      objectName,
IN DIVA_STRING      Category,
IN string           options
);
```

- `objectName`
Name of the object.

- `Category`
The Collection assigned to the object when it was archived.
- `options`
TBD

Return Values

One of the following `DIVA_STATUS` constants defined in `DIVAapi.h`:

- `DIVA_OK`
The job was correctly submitted and accepted by the Manager.
- `DIVA_ERR_NOT_CONNECTED`
No connection is open.
- `DIVA_ERR_SYSTEM_IDLE`
The DIVA system is no longer able to accept connections and queries.
- `DIVA_ERR_BROKEN_CONNECTION`
The connection with the Manager was broken.
- `DIVA_ERR_TIMEOUT`
The timeout limit was reached before communication with the Manager could be performed. The timeout duration is set by the `DIVA_API_TIMEOUT` variable and equals one hundred-eighty (180) seconds by default.
- `DIVA_ERR_UNKNOWN`
An unknown status was received from the Manager.
- `DIVA_ERR_INTERNAL`
An internal error was detected by the Manager or by the API.