

**Oracle® DIVArchive**  
C++ API Reference Manual  
Release 7.4  
**E73127-03**

September 2016

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Primary Author: Lou Bonaventura

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

---

---

# Contents

<b>Preface</b> .....	ix
Audience .....	ix
Documentation Accessibility .....	ix
Related Documents .....	ix
Conventions .....	ix
 <b>1 Introduction</b>	
<b>Oracle DIVArchive C++ API Overview</b> .....	1-1
<b>New and Enhanced Features and Functionality</b> .....	1-2
<b>Compilers</b> .....	1-2
Visual C++ Compiler on Windows .....	1-2
Supported Platforms .....	1-2
Supported Compilers .....	1-3
API Library Options .....	1-3
API Compilation .....	1-3
Initiator Sample Program API Usage .....	1-4
C++ Compiler on Linux .....	1-4
Supported Platforms .....	1-4
API Compilation .....	1-4
<b>Using the DIVArchive API in Multithreaded Applications</b> .....	1-5
<b>Managing Connections</b> .....	1-5
<b>Using Unicode Strings in the DIVArchive API</b> .....	1-5
<b>DIVArchive Release Compatibility</b> .....	1-6
<b>Alternate APIs</b> .....	1-6
 <b>2 API Use and Operations</b>	
<b>Session Management Commands</b> .....	2-1
DIVA_getApiVersion .....	2-1
Synopsis .....	2-1
DIVA_connect .....	2-1
Synopsis .....	2-1
Return Values .....	2-2
DIVA_disconnect .....	2-3
Synopsis .....	2-3
Multithreaded Applications .....	2-3

Return Values .....	2-3
<b>Requests and Commands .....</b>	<b>2-4</b>
DIVA_addGroup .....	2-4
Synopsis.....	2-4
Return Values .....	2-4
DIVA_archiveObject.....	2-5
Synopsis.....	2-5
Return Values .....	2-8
DIVA_associativeCopy .....	2-8
Synopsis.....	2-9
Return Values .....	2-9
DIVA_cancelRequest .....	2-10
Synopsis.....	2-11
Return Values .....	2-11
DIVA_changeRequestPriority .....	2-11
Synopsis.....	2-11
Return Values .....	2-12
DIVA_copyToGroup and DIVA_copy.....	2-13
Synopsis.....	2-13
Return Values .....	2-14
DIVA_copyToNewObject .....	2-15
Synopsis.....	2-15
Return Values .....	2-18
DIVA_deleteGroup .....	2-19
Synopsis.....	2-19
Return Values .....	2-19
DIVA_deleteInstance .....	2-20
Synopsis.....	2-20
Return Values .....	2-21
DIVA_deleteObject .....	2-22
Synopsis.....	2-22
Return Values .....	2-23
DIVA_ejectTape.....	2-24
Synopsis.....	2-24
Return Values .....	2-25
DIVA_enable_Automatic_Repack.....	2-26
Synopsis.....	2-26
Return Values .....	2-26
DIVA_getArchiveSystemInfo.....	2-27
Synopsis.....	2-27
Return Values .....	2-31
DIVA_getArrayList.....	2-31
Synopsis.....	2-31
Return Values .....	2-33
DIVA_getFinishedRequestList.....	2-33
Synopsis.....	2-34
Return Values .....	2-35

DIVA_getFilesAndFolders.....	2-35
Synopsis.....	2-35
Return Values .....	2-38
DIVA_getGroupsList.....	2-38
Synopsis.....	2-39
Return Values .....	2-39
DIVA_getObjectDetailsList.....	2-40
Synopsis.....	2-40
Return Values .....	2-47
Use with the DIVAnet Access Gateway .....	2-48
Use and Recommended Practices.....	2-48
Recommended Practices for Continuous Updates Notification Design Pattern (No Media Filter) 2-50	
DIVA_getObjectInfo .....	2-52
Synopsis.....	2-52
Return Values .....	2-52
DIVA_getPartialRestoreRequestInfo.....	2-53
Synopsis.....	2-53
Return Values .....	2-54
DIVA_getRequestInfo .....	2-54
Synopsis.....	2-54
Return Values .....	2-57
Additional_Info .....	2-58
DIVA_getSourceDestinationList.....	2-59
Synopsis.....	2-59
Return Values .....	2-61
DIVA_getStoragePlanList .....	2-61
Synopsis.....	2-61
Return Values .....	2-61
DIVA_getTapeInfo.....	2-62
Synopsis.....	2-62
Return Values .....	2-63
DIVA_insertTape .....	2-63
Synopsis.....	2-64
Return Values .....	2-65
DIVA_linkObjects .....	2-65
Synopsis.....	2-65
Return Values .....	2-66
DIVA_lockObject.....	2-66
Synopsis.....	2-66
Return Values .....	2-67
DIVA_multipleRestoreObject.....	2-67
Synopsis.....	2-67
Return Values .....	2-69
DIVA_partialRestoreObject .....	2-70
Synopsis.....	2-73
Return Values .....	2-80

DIVA_release .....	2-81
Synopsis.....	2-81
Return Values .....	2-81
DIVA_require .....	2-82
Synopsis.....	2-82
Return Values .....	2-82
DIVA_restoreInstance .....	2-83
Synopsis.....	2-83
Return Values .....	2-85
DIVA_restoreObject.....	2-86
Synopsis.....	2-86
Return Values .....	2-88
DIVA_transcodeArchive .....	2-89
Synopsis.....	2-89
Return Values .....	2-92
DIVA_transferFiles .....	2-92
Synopsis.....	2-93
Return Values .....	2-94
DIVA_unlockObject.....	2-94
Synopsis.....	2-94
Return Values .....	2-95

### **3 Using the DIVArchive API with Oracle DIVAnet**

What is DIVAnet? .....	3-1
DIVArchive API Support.....	3-1
Input Parameters .....	3-2
Return Parameters .....	3-2
Return Codes.....	3-2
getObjectDetailsList Call.....	3-2

### **A Appendix**

List of Authorized Special Characters in DIVArchive .....	A-1
Maximum Allowed Number of Characters.....	A-2
API Static Constant Values .....	A-2

### **Glossary**



## List of Tables

1-1	Unicode Strings .....	1-5
2-1	DIVA_getObjectDetailsList Function Values.....	2-47
A-1	Special Authorized Characters in DIVArchive.....	A-1
A-2	API Static Constants .....	A-3



---

---

# Preface

This document contains a detailed description of the Oracle DIVArchive and Oracle DIVAnet C++ Application Programmer's Interface (API).

## Audience

This document assists System Administrators and API Application Developers with development and deployment of applications interacting with DIVArchive and DIVAnet.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

## Related Documents

For more information refer to the *DIVArchive*, *DIVAnet*, *Java API*, *WS API*, and the C++ *Standard Template Library* (STL) documentation on the Oracle OTN.

## Conventions

The following text conventions are used in this document:

Convention	Meaning
<b>boldface</b>	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.



---

# Introduction

DIVArchive 7.4 enables interoperability among systems, helping to ensure long-term accessibility to valued content, and keeping up with evolving storage technologies.

The architecture of DIVArchive allows the integration of many different types of servers and technologies, for example Broadcast Video Servers, Storage Area Networks, and Enterprise Tape Libraries.

## Oracle DIVArchive C++ API Overview

The main Oracle DIVArchive API is written in the C++ programming language. All of the definitions are contained in the `include` file named `DIVAapi.h`. In this document, parameters 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 members<sup>1</sup>. Interfaces described in this document show only data members, not constructors or destructors.

The DIVArchive and DIVAnet 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 DIVArchive API is not supported under the Solaris operating system.

---

DIVArchive 7.4 does not currently 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 will be generated.

- `VerifyFollowingArchive`
- `VerifyFollowingRestore`
- `DeleteOnSource`

---

<sup>1</sup> The operators `new` and `delete` are not considered function members.

- DeleteFile
- getObjectListbyFileName
- The getObjectInfo and getObjectDetailsList will only return a single file

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

## New and Enhanced Features and Functionality

The following new and enhanced features and functionality are included in DIVArchive 7.4:

- DIVA\_getObjectInfo

The file list now contains empty files (that is, files of size 0 bytes). Client applications developed against API releases before release 7.4 will now receive empty files in the file list that accompanies an *Object Info* message.
- DIVA\_getObjectDetailsList

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 API releases before release 7.4 will now receive empty files in the file list that accompanies a *Details List* message.
- DIVA\_getFilesAndFolders
  - 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.

## Compilers

The following sections cover the supported API compilers.

### Visual C++ Compiler on Windows

These section describe using the Visual C++ compiler on the Windows operating system.

#### Supported Platforms

There are two separate variants of the DIVArchive 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 DIVArchive API for Windows is supported on the following Windows releases:

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2008
- Microsoft Windows Server 2008 x64
- Microsoft Windows Server 2008 R2

## Supported Compilers

The DIVArchive 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)

## API Library Options

The 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

## 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

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

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

### Initiator Sample Program API Usage

The Initiator program is included with the DIVArchive API and is an example of the API usage. This is a command line program that uses the API to send requests and get data from DIVArchive. 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)

## C++ Compiler on Linux

These sections describe using the C++ compiler on the Linux operating system platform.

### Supported Platforms

The DIVArchive API for Linux is supported on Oracle Linux. The API was built with the C++ compiler and Oracle Solaris Studio library. The following list identifies the supported CC release and Oracle Solaris Studio library release.

- Oracle Linux 7 x86\_64 (64-bit) operating system
- Oracle Solaris Studio 12.4 library

The following command returns the CC release level:

```
[root@LinuxBuildVM /]# CC -V
CC: Sun C++ 5.13 Linux_i386 2014/10/20
```

The DIVArchive API *may* work on other Linux platforms; however it is only officially validated in the environment described here. Support for the older release previously built on SuSe Linux 9.0 is discontinued in DIVArchive 7.4. *For all development projects, use of the latest release is strongly recommended.*

### API Compilation

The DIVArchive API is delivered with the x86\_64\_Release\_unicode shared library for the Linux platform. The release is located in the DIVA/api/lib directory. The library is built in *Release Mode* and does not contain symbolic information.

Header files that may be required to compile an application with the DIVArchive API libraries are delivered in the DIVA/api/include directory.

For reference, a sample application is provided in the DIVA/api/doc/CPPIInitiator directory along with its source code. The Visual Studio project file for Microsoft Windows, and sample makefiles for Linux platforms are also provided. Refer to the sample makefiles provided in the DIVA/api/doc/CPPIInitiator directory for platform-specific compiler and linker options.

## Using the DIVArchive API in Multithreaded Applications

The DIVArchive 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 DIVArchive functions may be called simultaneously by different threads. For example, one thread can call the `DIVA_archiveObject()` function while another one is calling `DIVA_getArchiveSystemInfo()`.

## Managing Connections

The number of connections to the Oracle DIVArchive Manager is limited by the Manager and set in the Manager configuration file. The default configuration is thirty connections, which includes GUI connections and all API connections. Once 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 request or command sent to the Manager. Whenever possible allow the connection to remain open for the lifetime of the session, or application.

---

## Using Unicode Strings in the DIVArchive API

The DIVArchive API (and other DIVArchive components) support wide character strings. Only 64-bit Unicode is delivered with the API. You must define the `_UNICODE` constant 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")
```

**Table 1–1 Unicode Strings**

Type	<code>_UNICODE</code> Not Defined	<code>_UNICODE</code> Defined
<code>DIVA_STRING</code>	<code>string</code>	<code>wstring</code>
<code>DIVA_CHAR</code>	<code>char</code>	<code>wchar_t</code>

## DIVArchive Release Compatibility

DIVArchive and Oracle DIVAnet are backward compatible with all earlier releases of the DIVArchive C++ API. Therefore, DIVArchive C++ API 7.3.x is compatible with any DIVArchive release 7.3 and later.

Any new features added to DIVArchive after the release of the C++ API in use will not be available; the client system must be upgraded to the latest release to use all features.

## Alternate APIs

The API described in this document is for use with applications implemented in C++. However, the following additional APIs are available:

- **Oracle DIVArchive Java API:** A set of libraries, samples and documentation for use with applications implemented in Java. See the *Oracle DIVArchive Java API Readme* in the *Oracle DIVArchive Additional Features* Documentation library on OTN for Java API document location information.
- **Oracle DIVArchive Web Services (WS) API:** A set of interface definition files and documentation for universal use by applications supporting Web Services communications. See the *Oracle DIVArchive WS API Reference* and the *Oracle DIVArchive WS API User's Guide* in the *Oracle DIVArchive Additional Features* documentation library on OTN for more information.



---

## API Use and Operations

This chapter discusses connection management, requests, and commands.

### 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\_connect

Opens a connection with the DIVArchive 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 DIVArchive Manager.

**portNumber**

The port on which the DIVArchive Manager is listening. 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 DIVArchive 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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_SYSTEM\_IDLE**

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

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_NO\_ARCHIVE\_SYSTEM**

There was a problem when establishing a connection with the specified DIVArchive 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 DIVArchive 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 "DIVApi.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 DIVArchive 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 `DIVApi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_DISCONNECTING**

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

Also see ["DIVA\\_connect"](#).

## Requests and Commands

The following sections discuss all of the available API commands for use in your application.

### DIVA\_addGroup

This function adds a new 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 group to be added.

#### **associatedSet**

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

#### **comment**

A text description of the new group.

#### **toBeRepacked**

If true, tapes belonging to this 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 group. The value can be **DIVA\_MEDIA\_FORMAT\_LEGACY** or **DIVA\_MEDIA\_FORMAT\_AXF**. See information on media formats in the ["Glossary"](#).

#### Return Values

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

#### **DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system can no longer accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive Manager could be performed. You set the timeout duration 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_GROUP\_ALREADY\_EXISTS**

The specified group already exists.

## DIVA\_archiveObject

Submits an archive request to the DIVArchive Manager. This function returns as soon as the Manager accepts the request. 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          objectCategory,
    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 category of the object to be archived.

**source**

The name of the source (for example, the video server, browsing server, and so on). This name must be known to the DIVArchive 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 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 Storage Plan (SP).

**SP Name**

Provide a Storage Plan Name (*SP 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.*

**filesPathRoot**

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

**filenamesList**

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

The following is for DIVArchive releases 7.1.2 and later *only*:

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:

**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 sources, disk sources, and standard ftp sources. *This feature is not available for complex objects.*

***priorityLevel***

The priority level for this request. 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 request.

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 to DIVArchive. These options supersede any options specified in the DIVArchive configuration database. Currently the possible values for *archiveOptions* are as follows:

**Null string**

A null string specifies no options.

**-r**

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

**-login**

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

**-pass**

The password used with **-login**.

The following is for DIVArchive releases 7.1.2 and later *only*:

**-gcinfilelist [*gcType*]**

Specifies that Genuine Checksum (GC) values are included in the file names list. The value of *gcType* must match the Manager's default checksum type as specified

in the DIVArchive configuration (MD5 by default). The GC values are then used to verify the transfer from the source.

**requestNumber**

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

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

DIVArchive can no longer accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive Manager could be performed. You set the timeout duration 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

The DIVArchive Manager or DIVArchive API detected an internal error.

**DIVA\_ERR\_INVALID\_PARAMETER**

The DIVArchive Manager did not understand a parameter value.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests reached the maximum allowed value. You set this variable 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 Source/Destination is unknown by the DIVArchive system.

**DIVA\_associativeCopy**

Submits a request for creating new instances in the group (specified by group). DIVArchive guarantees that these instances are stored sequentially on tapes:

- The request 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 request 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 request terminates (and is retried once) instead of spanning. The request terminates if the sum of the size of the objects to copy exceeds the capacity of every individual tape present in the library.

## 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 category pair.

### **groupName**

The name of the 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 request. 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 request.

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 parameters to the request.

### **requestNumber**

A number identifying the request.

## Return Values

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

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

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests 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 DIVArchive database.

**DIVA\_ERR\_SEVERAL\_OBJECTS**

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

**DIVA\_ERR\_OBJECT\_OFFLINE**

No available instance for this object. Tape instances are ejected and no DIVArchive 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 DIVArchive 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 request to be canceled. This parameter can be set to *DIVA\_ALL\_REQUESTS* to cancel all cancellable requests.

### ***options***

An optional string attribute for specifying additional parameters to the request.

## Return Values

One of the following **DIVA\_STATUS** constants defined in *DIVAapi.h*:

### **DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

### **DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

### **DIVA\_ERR\_INTERNAL**

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

### **DIVA\_ERR\_NO\_SUCH\_REQUEST**

The *requestNumber* identifies no request.

Also see "[DIVA\\_getRequestInfo](#)".

## DIVA\_changeRequestPriority

Submits a *Change Request Priority* request to the DIVArchive Manager. This function returns as soon as the Manager accepts the request. 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 request to be changed.

***priorityLevel***

The level of priority for this request. 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 parameters to the request.

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_NO\_SUCH\_REQUEST**

The *requestNumber* identifies no request.

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value has not been understood by the DIVArchive Manager.

Also see ["DIVA\\_getRequestInfo"](#).

**DIVA\_copyToGroup and DIVA\_copy**

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

The request will fail if the requested 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, DIVArchive will terminate the request if both instances cannot be written on two different tapes. A disk array can contain two instances of the same object; however DIVArchive will terminate the request 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 category 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 DIVArchive 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 request. 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 request.

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 parameters to the request.

***requestNumber***

A number identifying the request to be changed.

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value has not been understood by the DIVArchive Manager.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

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

The request will fail if the requested 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 object name (required).

**source.objectCategory**

The source object category (required).

**source.group**

The source object instance tape group or disk array. This is optional, however if specified DIVArchive will use this instance as the source.

**source.instanceID**

The Instance ID of the source object instance. This is optional, however if specified and not equal to *DIVA\_ANY\_INSTANCE*, DIVArchive will use this instance as the source. The *source.group* parameter will be ignored if *source.instanceID* is specified.

If both *source.group* and *source.instanceID* are omitted, DIVArchive 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).

**target.objectCategory**

The target object category (required).

**target.group**

See the following paragraph.

**target.instanceID**

This parameter is ignored.

Either the object name or category (or both) must be different from name or category of the source object. The request will fail if the target object already exists in DIVArchive.

**attrs**

The request attributes:

**attrs.priority**

The request priority (optional). If this is not explicitly set the default value is used. Possible values are zero (lowest) to one hundred (highest).

**attrs.qos**

Quality of Service (QOS) is not applicable to this request and this value is ignored.

**attrs.comments**

The target object's comments (optional). If no value is specified the source object's comments are inherited.



***attrs.options***

This request has no additional options and this is ignored.

***requestNumber***

The number identifying the request that is returned by DIVArchive.

```
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 object.

***objectCategory***

The category of the source object.

***objectMedia***

The tape group or disk array of the source object instance (optional). If specified (not empty), DIVArchive will use this instance as a source. Complex objects can only be saved to AXF formatted media types.

***objectInstanceID***

The Instance ID of the source object instance (optional). If specified and not equal to *DIVA\_ANY\_INSTANCE*, DIVArchive will use this instance as the source. The *objectMedia* parameter is ignored if *instanceID* is specified.

If both *objectMedia* and *instanceID* are not specified, DIVArchive will use the most suitable instance (providing the best performance) as the source.

***newObjectName***

The target object name.

***newObjectCategory***

The target object category. Either the object name or category (or both) must be different from name or category of the source object.

This request will fail if the target object already exists in DIVArchive.

***newObjectInstanceMedia***

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

***Name (of the 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 Storage Plan (SP).

**SP Name**

Provide a Storage Plan Name (*SP 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 object comments are inherited.

**priorityLevel**

Level of priority for this request. The possible values can be in the range zero to one hundred, and the **DIVA\_DEFAULT\_REQUEST\_PRIORITY** (use default request priority).

**options**

Optional string attribute for specifying additional parameters to the request.

**requestNumber**

The request number assigned to this request by DIVArchive.

**Return Values**

One of these **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests 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 DIVArchive 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 DIVArchive 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 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 group passed as an argument. You can only delete a group when the group is empty.

**Synopsis**

```
#include "DIVAapi.h"

IN DIVA_STRING          groupName
DIVA_STATUS DIVA_deleteGroup (
);
```

***groupName***

The name of the group to be deleted.

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`.

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_GROUP\_DOESNT\_EXIST**

The specified group does not exist.

**DIVA\_ERR\_GROUP\_IN\_USE**

The 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 category 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 2 or more instances, only one of the instances will be deleted.

***priorityLevel***

The level of priority for this request. 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 request.

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 parameters to the request.

***requestNumber***

A number identifying the request.

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests 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 DIVArchive database.

**DIVA\_ERR\_SEVERAL\_OBJECTS**

More than one object with the specified name exists in the DIVArchive 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 Request* to the DIVArchive Manager. The Manager deletes every instance of the object. This function returns as soon as the Manager accepts the request. 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 category 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 request. 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 request.

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 parameters to the request.

***requestNumber***

A number identifying the request.

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests 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 DIVArchive database.

**DIVA\_ERR\_SEVERAL\_OBJECTS**

More than one object with the specified name exists in the DIVArchive 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 DIVArchive database, but it is currently being archived.

See also ["DIVA\\_getRequestInfo"](#) and ["DIVA\\_deleteInstance"](#).

## DIVA\_ejectTape

Submits an *Eject Request* to DIVArchive. The request completes when the specified tapes are outside of the library.

If at least one of the tapes does not exist, is already ejected, or currently in use by another request, 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>    *vsnList,  
IN bool                   release  
IN DIVA_STRING            comment,  
IN int                   priorityLevel,  
OUT int                  *requestNumber  
);
```

***vsnList***

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 request. 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 request.

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 request.

**Return Values**

One of these **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager, or at least one of the barcodes 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 requests 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 DIVArchive Manager.

When the automatic repack scheduling is *enabled*, the schedule defined in the Control GUI is applied and tapes belonging to groups for which repack is allowed can be repacked according to the other automatic repack settings.

When the automatic repack scheduling is *disabled*, the schedule is ignored, all running automatic repack requests might be canceled (or not, according to other automatic repack settings), and no other automatic repack requests will be started until the automatic repack scheduling is turned on again (either from this API or from the Control GUI).

**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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

## DIVA\_getArchiveSystemInfo

Retrieves general information about the DIVArchive system.

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

The following parameters 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 DIVArchive Actor.

***actorAddress***

The DIVArchive 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 Library Storage Module name.

***lsmID***

This is the unique LSM ID.

***lsmlsAvailable***

This is true if the LSM identified by the preceding *lsmID* parameter is available for DIVArchive.

***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* described later.

***drivesAvailable***

This is true if the identified drive is available for DIVArchive.

***status***

The status of DIVArchive.

***lib\_status***

This is ok if at least one ACS is online. See *lsmList* described later.

***totalNumberOfObjects***

The number of objects managed by this DIVArchive system.

***actorsDrivesList***

<DIVA\_ACTOR\_AND\_DRIVES\_DESC>

***lsmList***

&lt;DIVA\_LSM\_DESC&gt;

***drivesList***

&lt;DIVA\_DRIVE\_DESC&gt;

***numberOfBlankTapes***

The number of blank tapes in a Set associated with at least one 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 group in an ACS where DIVArchive 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 group available for DIVArchive, 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 requests.

***currentRequests***

The number of current requests.

***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 Configuration Utility.

***siteIpAddress***

The Manager IP Address.

***sitePort***

The port number where the Manager is listening.

***firstUsedRequestId***

The first request ID used by the current Manager session. This value is -1 if no requests were processed.

***lastUsedRequestId***

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

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_getArrayList**

The purpose of this function is to provide a list of arrays and disks associated with the arrays in the DIVArchive system. It also returns arrays without any disks associated with them.

**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

class DIVA_ARRAY_DESC {
public:
    DIVA_STRING          arrayDesc;
    DIVA_STRING          arrayName;
```

```
int                numberOfDisk;
int                mediaFormatId;
vector<DIVA_DISK_ARRAY> *arrayDiskList;
};

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;
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**. See information on media formats in the "[Glossary](#)".

***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\_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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_getFinishedRequestList**

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

Use this function as follows:

If the list of requests 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 requests 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 requests. Requests 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 requests 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 request 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, DIVArchive 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 request was correctly submitted and accepted by the DIVArchive Manager.

### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

### **DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

### **DIVA\_ERR\_INTERNAL**

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

## DIVA\_getFilesAndFolders

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

You 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. You 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 category 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` parameters.

***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 only.

**DIVA\_LIST\_TYPE\_FOLDERS\_ONLY**

This function will return folders only. *This is only valid for complex objects.*

**DIVA\_LIST\_TYPE\_FILES\_AND\_FOLDERS**

This function will return files and folders. *This is valid for complex objects only.*

```
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 DIVArchive 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 DIVArchive 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 request.

**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 category of the object.

```
class DIVA_FILE_FOLDER_INFO {
public:
    DIVA_STRING      fileOrFolderName;
    bool             isDirectory;
    __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 and not a file name.

**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 DIVArchive 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 DIVArchive 7.4 API includes the following enhancements to the return values for this call:

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

One of these **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_WARN\_NO\_MORE\_OBJECTS**

The end of the list was reached during the call.

**DIVA\_getGroupsList**

Returns the description of all groups.

## 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 group. The value can be **DIVA\_MEDIA\_FORMAT\_LEGACY**, **DIVA\_MEDIA\_FORMAT\_AXF**, or **DIVA\_MEDIA\_FORMAT\_AXF\_10**. See information on media formats in the ["Glossary"](#).

## Return Values

One of these **DIVA\_STATUS** constants defined in `DIVAapi.h`:

### **DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

### **DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

### **DIVA\_ERR\_INTERNAL**

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

See also ["DIVA\\_getObjectInfo"](#).

## DIVA\_getObjectDetailsList

The **DIVA\_getObjectDetailsList** is an API call to retrieve object information from the DIVArchive 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 DIVArchive 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.

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

**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 request 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 DIVArchive system name as configured in `manager.conf`.

***listPosition***

After the first and any subsequent call, the DIVArchive 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 category.

***UUID***

Universally Unique Identifier to uniquely identify each object created in DIVArchive across all Oracle customer sites. This does not include objects created using Copy As requests. An object created through a Copy As request will contain the same *UUID* as that of the source 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 library, 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 system used to archive the object.

***rootDirectory***

The root directory containing the object files on the *objectsource*.

***relatedRequests***

This is non-terminated requests.

***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.

***objectCategory***

This is the object category.

```
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 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 library, 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 library.

**DIVA\_RELEASED** - There is no need to have this instance present into the library.

```
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 library 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*.

```
class DIVA_ACTOR_INSTANCE_DESC {
public:
    int         instanceID;
    string      actor;
};
```

**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 API releases before release 7.4 will now receive empty files in the file list that accompanies a Details List message. Depending on the input parameters, the **DIVA\_getObjectDetailsList** function will return values as described in the following table.

**Table 2–1** *DIVA\_getObjectDetailsList Function Values*

List Type	Objects 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 <i>empty</i> , objects were <i>deleted</i> .  If the list of instances is <i>not empty</i> , objects were <i>created</i> or <i>updated</i> .
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.

**Table 2–1 (Cont.) DIVA\_getObjectDetailsList Function Values**

List Type	Objects List Type	Supported Detail Level	Return Value
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 the DIVAnet Access Gateway

All filters are applied at an object level as follows:

- If you request objects satisfying certain filter constraints, those constraints are applied to the object and not to individual instances of an object.
- If you specify an object name and category filter, the list will be filtered to contain only objects satisfying the specified object name and category.

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 requested media name filter.

---

**Note:** If an instance of an object is created or deleted, and you request all modified objects with a particular media name, 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

Oracle recommends that the DIVArchive 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 you want object information for modified objects.
6. Create a variable of Integer type to hold the suggested number of objects you want returned by the call.
7. Create list filtering variables of **DIVA\_CHAR[]** type to hold the object name, category and media filters.
8. Create a variable of Integer type to hold the level of detail you want returned.
9. Execute **DIVA\_GetObjectDetailsList** with the variables previously mentioned.
10. Use the data stored in the variable from Step 1 as needed by your application.
11. Copy the *listPosition* attribute of the call's output created in Step 1 into the *listPosition* variable created in Step 2.
12. Continue repeating steps 8, 9, and 10 until you no longer need to monitor DIVArchive.
13. All variables must be unallocated 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 DIVArchive 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, you must compare the instances of the duplicate object returned by successive calls to identify whether new information about the object is available and update your 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 DIVArchive with the **DIVA\_getObjectDetailsList** call using the ID from the previous call. However, the call frequency must be reduced after you receive an empty list. This reduces the load on the DIVArchive 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 DIVArchive

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. Oracle 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, Oracle 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 DIVArchive API. The client application can use the internal database to continuously update the local database information with changes in the DIVArchive 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 parameters must be set (object name and category).

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 of time between calls to allow other DIVArchive requests to process.
3. Repeat Steps 1 and 2 for the course of execution to keep the internal database synchronized with DIVArchive database.
4. If none of the objects in DIVArchive 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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

An internal error was detected by the DIVArchive Manager or by the DIVArchive 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 DIVArchive 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 DIVArchive 7.4 file list can contain empty files (that is, files of size 0 bytes). Client applications developed against API releases before release 7.4 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 an actor instance) and its *string actor* field now contains the array name instead of an Actor name.

### 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 category 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 parameters to the request.

***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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_OBJECT\_DOESNT\_EXIST**

The specified object does not exist in the DIVArchive Database.

**DIVA\_ERR\_SEVERAL\_OBJECTS**

More than one object with the specified name exists in the DIVArchive Database.

See also "[DIVA\\_archiveObject](#)", "[DIVA\\_restoreObject](#)", and "[DIVA\\_deleteObject](#)".

**DIVA\_getPartialRestoreRequestInfo**

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

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

- The request number to be queried must be a partial file restore request that has been successfully completed.
- The format specified in the partial file restore request must be a timecode type. This command is therefore not valid when the format of the request 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 Oracle DIVArchive Partial File Restore request to be queried.

***fileList***

List of the files of an object that have been partially restored. Each structure contains the source file name, a vector of the offsets used for the transfer, and a destination 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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_NO\_SUCH\_REQUEST**

The `requestNumber` identifies no request

**DIVA\_ERR\_INVALID\_PARAMETER**

The `requestNumber` identifies no completed partial file restore request.

See also "[DIVA\\_partialRestoreObject](#)" and "[DIVA\\_getRequestInfo](#)".

**DIVA\_getRequestInfo**

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

**Synopsis**

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

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

***requestNumber***

Identifies the queried request.

***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
    DIVA_REQUEST_STATE requestState;
    DIVA_REQUEST_STATE
    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 DIVArchive request 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 request from zero to one hundred percent if the *requestState* is **DIVA\_TRANSFERRING** or **DIVA\_MIGRATING**.

***abortionReason***

The reason the request 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 request was submitted. This is UTC time in seconds (that is, seconds since January 1, 1970).

***completionDate***

The date and time the request completed. This is UTC time in seconds and will be -1 if the request 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**

Request 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**

Source/Destination trouble

**DIVA\_AR\_RESOURCES**

Resource attribution trouble

**DIVA\_AR\_LIBRARY**

Library trouble

**DIVA\_AR\_PARAMETERS**

Incorrect request parameters

**DIVA\_AR\_UNKNOWN**

Unknown code

**DIVA\_AR\_INTERNAL**

Internal DIVArchive 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 category of the object.

**Return Values**

One of these **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_NO\_SUCH\_REQUEST**

The `requestNumber` identifies no request

**Additional\_Info**

The `Additional_Info` field of the **DIVA\_REQUEST\_INFO** structure can contain one or more of the following depending on the request type:

**MOB ID**

`MOB ID` is a unique object identifier generated and used by AVID software. The DIVArchive 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 request 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 DIVArchive installation.

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

```
<ADDITIONAL_INFO
xmls="http://www.fpdigital.com/divarchive/additionalInfoRequestInfo/v1.0>"
<Object>
  <Name>Object Name</Name>
  <Category>category</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 request was a Multiple Restore. If the restore is OK for one of the destinations, 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 is as follows:

```

<ADDITIONAL_INFO
xmls="http://www.fpdigital.com/divarchive/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 request 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 DIVArchive API, *DIVA\_GetRequestInfo* must be called. If the request is completed, the new ClipID will be in the Additional Request Information field as follows:

```

<ADDITIONAL_INFO
xmls="http://www.fpdigital.com/divarchive/additionalInfoRequestInfo/v1.0">
    <ClipID>98765</ClipID>
</ADDITIONAL_INFO>

```

## DIVA\_getSourceDestinationList

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

### Synopsis

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

```

DIVA_STATUS          DIVA_getSourceDestinationList (
IN string             options;
OUT vector<DIVA_SOURCE_DESTINATION_LIST> *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 production system name.

***server\_RootPath***

The server root path.

***server\_SourceType***

The server source type.

**Return Values**

One of these **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_getStoragePlanList**

This function returns the list of Storage Plan Names that are defined in the DIVArchive 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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

An internal error was detected by the DIVArchive Manager or by the DIVArchive 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:

$$\text{last\_written\_block} / \text{total\_block\_count}.$$
**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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_TAPE\_DOESNT\_EXIST**

There is no tape associated with the given barcode.

**DIVA\_insertTape**

Submits an Insert request to DIVArchive. This request completes when the operator has entered the requested tapes into the library. 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 request. 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 request.

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 request.



## Return Values

One of these **DIVA\_STATUS** constants defined in `DIVAapi.h`:

### **DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

### **DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

### **DIVA\_ERR\_INTERNAL**

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

### **DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

### **DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests 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.

***parentCategory***

The parent object category.

***childName***

The child object name.

***childCategory***

The child object category.

***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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_OBJECT\_ALREADY\_EXISTS**

An object with this name and category already exists in the DIVArchive system.

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

An internal error was detected by the DIVArchive Manager or by the DIVArchive 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 category 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 request was correctly submitted and accepted by the DIVArchive Manager.

### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

### **DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

### **DIVA\_ERR\_INTERNAL**

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

## DIVA\_multipleRestoreObject

Submits an object Restore request to the DIVArchive Manager using several destinations. DIVArchive Manager chooses the appropriate instance to be restored. This function returns as soon as the Manager accepts the request.

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

If `DIVA_MultipleRestoreObject()` is launched with a single destination, 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 category 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 destinations (for example, a video server or browsing server) where object files can be restored. The names must be known by the DIVArchive configuration description.

A root folder where the object files will be placed is associated with each destination. 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 request. 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 request.

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 DIVArchive to the destination. These options supersede any options specified in the DIVArchive configuration database. Currently the possible values for *restoreOptions* are:

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

#### ***requestNumber***

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

### **Return Values**

One of these **DIVA\_STATUS** constants defined in `DIVAapi.h`:

#### **DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

#### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

#### **DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system cannot accept connections and queries.

#### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

#### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

#### **DIVA\_ERR\_INTERNAL**

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

#### **DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests 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 DIVArchive database.

**DIVA\_ERR\_OBJECT\_OFFLINE**

There is no inserted instance in the library and no Actor could provide a disk instance.

**DIVA\_ERR\_SEVERAL\_OBJECTS**

More than one object with the specified name exists in the DIVArchive 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 Source/Destination is unknown by the DIVArchive 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 request to the DIVArchive Manager and the Manager chooses the appropriate instance to be restored. This function returns as soon as the Manager accepts or rejects the request. To check that the operation was successful the application must call the `DIVA_getRequestInfo()` function.

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 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

DIVArchive supports four types of Partial Restore. The type implemented is determined by the *format* parameter in the request.

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, you can extract bytes 1 to 2000 (the first 2000 bytes of the file), or byte 5000 to the end of the file (or both) and store them 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 request. Create a **DIVA\_OFFSET\_SOURCE\_DEST** object (in the *fileList* parameter of the request). In the object you must specify the *sourceFile* in the archive and name the output file (*destFile*). 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, you could extract from 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) and 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 request 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 request 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 request. DIVArchive 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 request. 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* parameters 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 DIVArchive you can extract multiple files and directories in the same request. 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 request must be populated with the **DIVA\_FORMAT\_FOLDER\_BASED** value. Create a **DIVA\_OFFSET\_SOURCE\_DEST** object in the *fileList* parameter of the request. 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 you extract frame 10 through frame 15 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 you have 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, you cannot specify Frame -1 to -1 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* parameters should not be specified for the DPX Partial Object restore type.

---

---

To issue a DPX Partial Restore you populate the format field in the request with the value **DIVA\_FORMAT\_DPX**. Create a **DIVA\_OFFSET\_SOURCE\_DEST** object in the *fileList* parameter of the request. In this object, you 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 request, another **DIVA\_OFFSET\_SOURCE\_DEST** object should be added to the request in the same manner.

The actual file name may, or may not, match the frame number in DIVArchive. During the restore process DIVArchive 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.

You must 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 and is typically alphanumeric. For example, NTFS DISK Source/Destinations order files and folders case insensitively as a general rule except where diacritical marks such as ' , ` ^ , and so on are applied.

By default, when DIVArchive 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 you want all of the subdirectories of a given directory processed first, followed by the files in the directory, or you might want all files processed first and then subdirectories. In DIVArchive 7.4 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.

DIVArchive does not perform any special audio handling for DPX media other than what might be embedded in DPX files themselves. DIVArchive 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***

Category assigned to the object when it was archived. This parameter can be a null string, which can 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 file name, a vector of offset pairs, and a destination file name. The same source file can be used in several structures, but destination files must be unique. A file present in the DIVArchive object cannot be in any structure or it won't be restored.

***destination***

Destination (for example, a video server or browsing server) to put the object files. This name must be known by the DIVArchive configuration description.

***filePathRoot***

The root folder on the destination 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**

Use cache restore only.

**DIVA\_QOS\_DIRECT\_ONLY**

Use direct restore only.

**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 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 request. 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 request.

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 DIVArchive to the destination. These options supersede any options specified in the DIVArchive configuration database. Currently the possible values for *restoreOptions* are:

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

### ***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 DIVArchive creates the destination file by concatenating every corresponding extract.

#### **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 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). DIVArchive prefers the MSS Source/Destination type for creating this clip.

The *fileList* vector parameter must contain only one **DIVA\_OFFSET\_SOURCE\_DEST** element. The *offsetVector* vector must contain only one **DIVA\_OFFSET\_PAIR** element. The **DIVA\_OFFSET\_SOURCE\_DEST** element must be associated with the header file only. The destination name is also the header.

**DIVA\_FORMAT\_VIDEO\_OMNEON**

Offsets must be given as timecodes. This type of Partial File Restore partially restores 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* vector 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 partially restores Quantel clips that have been archived with a **QUANTEL\_QCP** Source/Destination 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* vector 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. You can set a recursive flag 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 refer to the first and last frame respectively.

#### ***requestNumber***

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

```
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 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 request. 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. 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\_MPEG\_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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

DIVArchive can no longer accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive Manager could be performed. You set the timeout duration 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

The DIVArchive Manager or DIVArchive API detected an internal error.

**DIVA\_ERR\_INVALID\_PARAMETER**

The DIVArchive Manager did not understand a parameter value.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests reached the maximum allowed value. You set this variable 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 DIVArchive database.

**DIVA\_ERR\_OBJECT\_OFFLINE**

There is no inserted instance in the library and no Actor could provide a disk instance.

**DIVA\_ERR\_SEVERAL\_OBJECTS**

More than one object with the specified name exists in the DIVArchive 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 (that is, being Archived, Restored, Deleted, and so on).

**DIVA\_ERR\_SOURCE\_OR\_DESTINATION\_DOESNT\_EXIST**

The specified Source/Destination is unknown by the DIVArchive 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 DIVArchive 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 at the Control GUI.

### 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 category 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 request was correctly submitted and accepted by the DIVArchive Manager.

#### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

#### **DIVA\_ERR\_SYSTEM\_IDLE**

DIVArchive can no longer accept connections and queries.

#### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

#### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive Manager could be performed. You set the timeout duration 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 DIVArchive Manager.

#### **DIVA\_ERR\_INTERNAL**

The DIVArchive Manager or DIVArchive API detected an internal error.

**DIVA\_ERR\_INVALID\_PARAMETER**

The DIVArchive Manager did not understand a parameter value.

**DIVA\_ERR\_OBJECT\_DOESNT\_EXIST**

The specified object does not exist in the DIVArchive 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 DIVArchive database.

See also "[DIVA\\_require](#)".

## DIVA\_require

Indicates to the DIVArchive 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 at the Control GUI.

### 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***

Category 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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

DIVArchive can no longer accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive Manager could be performed. You set the timeout duration 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

The DIVArchive Manager or DIVArchive API detected an internal error.

**DIVA\_ERR\_INVALID\_PARAMETER**

The DIVArchive Manager did not understand a parameter value.

**DIVA\_ERR\_OBJECT\_DOESNT\_EXIST**

The specified object does not exist in the DIVArchive 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 DIVArchive 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          filesPathRoot,
    IN DIVA_RESTORE_QOS     qualityOfService,
    IN int                  priorityLevel,
    IN DIVA_STRING          restoreOptions,
    OUT int                 *requestNumber
);
```

***objectName***

Name of the object to be restored.

***objectCategory***

Category 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 (for example, a video server or browsing server) where the object files will be restored. This name must be known by the DIVArchive configuration description.

***filePathRoot***

Root folder on the destination 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 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 request. 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 request.

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 DIVArchive to the destination. These options supersede any options specified in the DIVArchive 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 sources. This option obsoletes the **-gateway** option from earlier releases.

##### **-pass**

The password used with **-login**.

#### ***requestNumber***

A number identifying this request.

#### **Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

##### **DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

##### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

##### **DIVA\_ERR\_SYSTEM\_IDLE**

DIVArchive can no longer accept connections and queries.

##### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

##### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive Manager could be performed. You set the timeout duration 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 DIVArchive Manager.

##### **DIVA\_ERR\_INTERNAL**

The DIVArchive Manager or DIVArchive API detected an internal error.

##### **DIVA\_ERR\_INVALID\_PARAMETER**

The DIVArchive Manager did not understand a parameter value.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

Count of simultaneous requests 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 DIVArchive database.

**DIVA\_ERR\_SEVERAL\_OBJECTS**

More than one object with the specified name exists in the DIVArchive 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 Source/Destination is not known by the DIVArchive 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 request to the DIVArchive Manager and the Manager chooses the appropriate instance to be restored. This function returns as soon as the Manager accepts the request. To check that the operation was successful, the application must call the function `DIVA_getRequestInfo()`.

If the requested object is on media that is not available, the request 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***

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

***destination***

The destination (for example, a video server or browsing server) where the object files will be restored. This name must be known by the DIVArchive configuration description.

***filePathRoot***

Root folder on the destination 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 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.

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**

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**

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 request. 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 request.

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 DIVArchive to the destination. These options supersede any options specified in the DIVArchive 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 sources. This option obsoletes the **-gateway** option from earlier releases.

**-pass**

The password used with **-login**.

***requestNumber***

Request number assigned to this request. This number is used for querying the status or canceling this request.

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

DIVArchive can no longer accept connections and queries.



**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive Manager could be performed. You set the timeout duration 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

The DIVArchive Manager or DIVArchive API detected an internal error.

**DIVA\_ERR\_INVALID\_PARAMETER**

The DIVArchive Manager did not understand a parameter value.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests reached the maximum allowed value. You set this variable 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 DIVArchive database.

**DIVA\_ERR\_OBJECT\_OFFLINE**

There is no inserted instance in the library and no Actor could provide a Disk Instance.

**DIVA\_ERR\_SEVERAL\_OBJECTS**

More than one object with the specified name exists in the DIVArchive 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 Source/Destination is not known by the DIVArchive 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 request to the DIVArchive 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 DIVArchive.

### 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***

Category 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***

Category 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 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 Storage Plan (SP).

***SP Name***

Provide a Storage Plan Name (*SP 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 to DIVArchive. These options supersede any options specified in the DIVArchive configuration database. Currently the possible values for *archiveOptions* are:

```
-tr_archive_format FORMAT
```

Destination 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 DIVArchive 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 DIVArchive 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 request. 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 request.

Using a value either outside of the range of zero to one hundred, or predefined values yields a **DIVA\_ERR\_INVALID\_PARAMETER** error.

**requestNumber**

Request number assigned to this request. This number is used for querying the status or canceling this request.

**Return Values**

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

**DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

The DIVArchive system can no longer accept connections and queries.

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive Manager could be performed. You set the timeout duration 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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

**DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

**DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests reached the maximum allowed value. You set this variable in the `manager.conf` configuration file. The default value is three hundred.

**DIVA\_ERR\_OBJECT\_ALREADY\_EXISTS**

The specified object already exists in the DIVArchive 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 request to the DIVArchive Manager. The request will transfer files from a remote server (the source) to another remote server (the destination). This function returns as soon as the Manager accepts the request. 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 (for example, a video server or browsing server). This name must be known by the DIVArchive 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 (for example a video server or browsing server). This name must be known by the DIVArchive configuration description.

### **destinationPathRoot**

Root folder where the files will be placed at the destination.

### **priorityLevel**

The priority level for this request. 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 request.

Using a value either outside of the range of zero to one hundred, or predefined values yields a **DIVA\_ERR\_INVALID\_PARAMETER** error.

### **requestNumber**

Request number assigned to this request. This number is used for querying the status or canceling this request.

## Return Values

One of the following **DIVA\_STATUS** constants defined in `DIVAapi.h`:

### **DIVA\_OK**

The request was correctly submitted and accepted by the DIVArchive Manager.

### **DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

### **DIVA\_ERR\_SYSTEM\_IDLE**

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

### **DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

### **DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

### **DIVA\_ERR\_INTERNAL**

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

### **DIVA\_ERR\_INVALID\_PARAMETER**

A parameter value was not understood by the DIVArchive Manager.

### **DIVA\_ERR\_CANNOT\_ACCEPT\_MORE\_REQUESTS**

The count of simultaneous requests 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 Source/Destination is not known by the DIVArchive 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 category 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 request was correctly submitted and accepted by the DIVArchive Manager.

**DIVA\_ERR\_NOT\_CONNECTED**

No connection is open.

**DIVA\_ERR\_SYSTEM\_IDLE**

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

**DIVA\_ERR\_BROKEN\_CONNECTION**

The connection with the DIVArchive Manager was broken.

**DIVA\_ERR\_TIMEOUT**

The timeout limit was reached before communication with the DIVArchive 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 DIVArchive Manager.

**DIVA\_ERR\_INTERNAL**

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





---

## Using the DIVArchive API with Oracle DIVAnet

In addition to being able to connect to a DIVArchive system, the DIVArchive API connects to an Oracle DIVAnet system. This functionality enables applications to access content across multiple DIVArchive systems, possibly in different geographical locations. DIVAnet enables the content in each system to be retrieved and stored as if the DIVArchive sites together were one large archival system.

### What is DIVAnet?

DIVAnet provides a unified view of archived content across multiple, distributed DIVArchive systems. It facilitates the moving of content among DIVArchive sites, and from customer source and destination servers and disk. The purpose is for disaster recovery, content distribution, access control, performance, and content availability.

DIVAnet synchronizes asset information from each DIVArchive site, so that users always have an up-to-date inventory of where content is. DIVAnet uses this information to choose the best site for various requests, for example restores and copies. DIVAnet also provides access rules to limit the operations that users are permitted to perform.

DIVAnet 2.1 is a new release for compatibility with DIVArchive 7.4 Linux-based installations. DIVAnet 2.1 also runs on Windows-based systems. However, it is not backward compatible to releases before DIVArchive 7.3.1. You must use either DIVAnet 2.0 or Legacy DIVAnet (Release 1.0) when running DIVArchive releases earlier than DIVArchive 7.3.1.

The Legacy Oracle DIVAnet (Release 1.0) is still available for connecting DIVArchive systems with different software release levels, and releases before DIVArchive 7.3.1.

If you are operating a DIVArchive release earlier than 7.3.1, refer to the *Oracle DIVAnet Installation, Configuration, and Operations Guide* in the *Oracle DIVAnet 2.0 Documentation* library, or the appropriate Legacy DIVAnet documentation in the *Oracle DIVArchive Legacy library* (for DIVArchive releases 6.5 and 7.2).

### DIVArchive API Support

DIVAnet has partial support for the full DIVArchive API command set. Refer to the appropriate DIVAnet documentation for a complete list of supported API commands. DIVAnet will support client connections from DIVArchive API clients release 7.4 and earlier. New parameters or features added to the API after release 7.4 are not supported by DIVAnet. In general, a released version of DIVAnet can connect to newer releases of DIVArchive, and sometimes also can connect to older releases. This ability varies based on the specific release of DIVAnet.

## Input Parameters

Invoking DIVArchive API calls to a DIVAnet server is largely the same as invoking calls to DIVArchive. However, there are some differences. DIVAnet sometimes accepts additional information by using common DIVArchive API parameters in a slightly a different way.

For example, the DIVAnet Copy command (`CopyToGroup`) copies content from one DIVArchive system to another. DIVAnet needs to know, at a minimum, what the target DIVArchive site is. This information can be provided in multiple ways, for example you can prefix the *target\_sitename* to the media provided in the call (for example, `sitename2_mytapegroup`). Refer to the appropriate DIVAnet documentation for more information on specifying DIVAnet-specific information in DIVArchive API calls.

## Return Parameters

A DIVAnet system sometimes returns API information that is slightly different than you would typically see in a DIVArchive system. For example, the DIVAnet `getObjectInfo()` call returns information about an archived object across all DIVArchive sites. To distinguish which site is which, the source site name is prefixed to the media of each archived object instance returned in the call. For example, an object on *sitename2* that is stored on *mytapegroup* would have a media value of `sitename2_mytapegroup`.

Another example of a slight difference is the object instance ID. DIVArchive has a unique instance ID for each instance of an archived object (starting at zero and incrementing by one for each new instance). However, this value is not unique across DIVArchive sites. DIVAnet applies a simple algorithm to the instance ID to make it unique across sites (but not across objects). The unique DIVAnet instance IDs for an object can be queried by making a DIVAnet `getObjectInfo()` call.

The Request ID returned by each DIVAnet request does not necessarily correspond to a DIVArchive Request ID. Refer to the appropriate DIVAnet documentation for more information.

## Return Codes

DIVAnet will return **DIVA\_ERR\_ACCESS\_DENIED** if a user or connection does not have permission to perform a particular action. DIVArchive does not return this code. DIVAnet can possibly refuse an API connection altogether because of configured permissions. DIVArchive will accept the connection if it hasn't run out of available connections. There are cases where DIVAnet will choose to acknowledge a request with **DIVA\_OK** and then subsequently return an error (for example, an *Invalid Media* error). DIVArchive will simply reject the request with the **DIVA\_ERR\_INVALID\_PARAMETER** error.

## getObjectDetailsList Call

The `GetObjectDetailsList()` command retrieves a list of objects from each site. DIVAnet retrieves the object information directly from each DIVArchive system, one site at a time, in a round-robin fashion. It returns one batch per site to the initiator. The initiator must keep calling `GetObjectDetailsList()` with the same query parameters - passing all received list position data as input to the next call.

If an object is returned in one batch, the initiator can possibly receive the same object again in the next batch (for the second site). This makes `GetObjectDetailsList()` different from `GetObjectInfo()`, which returns information from all sites in one call.

The query parameters and time ranges queried in each batch are specific to each site. It is possible that if *Site1* contains many objects in a given query (and *Site2* does not). The batches from *Site2* that are near the end of the calling sequence might be completely empty.

Keep calling `GetObjectDetailsList()`, ignoring empty batches until the call returns either a status of **DIVA\_WARN\_NO\_MORE\_OBJECTS** or an error. All DIVArchive sites in the DIVAnet network must be online for `GetObjectDetailsList()` to succeed. If, for any reason, an error is returned before the list has been fully returned the entire calling sequence must be repeated.

Other details of the DIVArchive `GetObjectDetailsList()` call remain in effect for the DIVAnet version. For example, while the batches returned are ordered by time, the order of entries within each batch is not guaranteed. Although duplicate objects will not appear within a batch, the same object may appear in the next batch - the likelihood of this occurrence increases when you use the **MODIFIED\_SINCE** parameter.

If an object has been deleted and subsequently re-added, `GetObjectDetailsList()` will return one record for every time this has occurred for as long as DIVArchive retains the records.

To continuously monitor DIVAnet for new objects and instances, you can continue to call `GetObjectDetailsList()` even after it has returned a status of **DIVA\_WARN\_NO\_MORE\_OBJECTS**. To do this you must provide the exact same query information (passing all received list position data into the next call) to get any new updates since you last called it. If an error occurs, you must use the exact same list position that was received on the last successful call.

Refer to the appropriate DIVAnet documentation for more information on specific DIVArchive API calls.



## Appendix

The following appendixes sections include authorized special characters, character limitations, and static constants.

### List of Authorized Special Characters in DIVArchive

The following table lists the special characters that can be used in DIVArchive and in which fields they are valid.

**Table A–1** *Special Authorized Characters in DIVArchive*

Character	Name	Category	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
%	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
^	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
&	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
*	No	No	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

**Table A–1 (Cont.) Special Authorized Characters in DIVArchive**

Character	Name	Category	Source	Media	Path	File	Comments	Options
:	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
;	Yes	Yes	Yes	Yes	Yes <sup>1</sup>	Yes	Yes	Yes
"	Yes	Yes	Yes	Yes	No	Yes	Yes	No
'	Yes	Yes	No	No	Yes <sup>1</sup>	Yes	Yes	Yes
<	Yes	Yes	Yes	Yes	No	Yes	Yes	No
,	Yes	Yes	Yes	Yes	Yes <sup>1</sup>	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

<sup>1</sup> In a Windows environment, the file and folder name restrictions depend on the file system restrictions. File and folder names cannot solely consist of one or more spaces, and cannot contain a double-quote.

## Maximum Allowed Number of Characters

The maximum allowable number of characters are as follows:

### **Name**

192 maximum characters

### **Category**

96 maximum characters

### **Source**

96 maximum characters

### **Media**

96 maximum characters

### **Path and File Name**

1536 maximum characters per folder or per file

### **Comments**

4000 maximum characters

### **Options**

768 maximum characters

## API Static Constant Values

The following table identifies the values for each of the API static constants.

**Table A-2 API Static Constants**

Static Constant Name	Description	Value
DIVA_OK	The request was correctly submitted and accepted by the DIVArchive Manager.	1000
DIVA_ERR_UNKNOWN	An unknown status was received from the DIVArchive Manager.	1001
DIVA_ERR_INTERNAL	An internal error was detected by the DIVArchive Manager or the DIVArchive API.	1002
DIVA_ERR_NO_ARCHIVE_SYSTEM	Problem when establishing a connection with the specified DIVArchive system.	1003
DIVA_ERR_BROKEN_CONNECTION	The connection with the DIVArchive Manager was broken.	1004
DIVA_ERR_DISCONNECTING	Problem when disconnecting. The connection is still considered to be open.	1005
DIVA_ERR_ALREADY_CONNECTED	A connection is already open.	1006
DIVA_ERR_WRONG_VERSION	Release level of the API and the Manager are not compatible.	1007
DIVA_ERR_INVALID_PARAMETER	A parameter value was not understood by the DIVArchive Manager.	1008
DIVA_ERR_OBJECT_DOESNT_EXIST	The specified object does not exist in the DIVArchive database.	1009
DIVA_ERR_SEVERAL_OBJECTS	More than one object with the specified name exists in the DIVArchive database.	1010
DIVA_ERR_NO_SUCH_REQUEST	The requestNumber identifies no request.	1011
DIVA_ERR_NOT_CANCELABLE	The request is at the point where it cannot be canceled.	1012
DIVA_ERR_SYSTEM_IDLE	The DIVArchive System is no longer able to accept connections and queries.	1013

**Table A–2 (Cont.) API Static Constants**

Static Constant Name	Description	Value
DIVA_ERR_WRONG_LIST_SIZE	The list size is zero or larger than the maximum allowable value.	1014
DIVA_ERR_LIST_NOT_INITIALIZED	The specified list was not properly initialized. Initialization call was not executed.	1015
DIVA_ERR_OBJECT_ALREADY_EXISTS	An object with this name and category already exists in the DIVArchive system.	1016
DIVA_ERR_GROUP_DOESNT_EXIST	The specified group does not exist.	1017
DIVA_ERR_SOURCE_OR_DESTINATION_DOESNT_EXIST	The specified source or destination does not exist.	1018
DIVA_WARN_NO_MORE_OBJECTS	The end of the list was reached during the call.	1019
DIVA_ERR_NOT_CONNECTED	No open connection.	1020
DIVA_ERR_GROUP_ALREADY_EXISTS	The specified group already exists.	1021
DIVA_ERR_GROUP_IN_USE	The group contains at least one object instance.	1022
DIVA_ERR_OBJECT_OFFLINE	There is no inserted instance in the library and no Actor could provide a disk instance.	1023
DIVA_ERR_TIMEOUT	The timeout limit was reached before communication with the DIVArchive Manager could be performed. The timeout duration is set by the <i>DIVA_API_TIMEOUT</i> variable and equals one hundred-eighty (180) seconds by default.	1024
DIVA_ERR_LAST_INSTANCE	<i>DIVA_deleteObject()</i> must be used to delete the last instance of an object.	1025
DIVA_ERR_PATH_DESTINATION	The specified destination path is invalid.	1026



**Table A–2 (Cont.) API Static Constants**

Static Constant Name	Description	Value
DIVA_ERR_INSTANCE_DOESNT_EXIST	Instance specified for restoring this object does not exist.	1027
DIVA_ERR_INSTANCE_OFFLINE	Instance specified for restoring this object is ejected, or the Actor owning the specified disk instance is unavailable.	1028
DIVA_ERR_INSTANCE_MUST_BE_ON_TAPE	The specified instance is not a tape instance.	1029
DIVA_ERR_NO_INSTANCE_TAPE_EXIST	No tape instance exists for this object.	1030
DIVA_ERR_OBJECT_IN_USE	The object is currently in use (being Archived, Restored, Deleted, and so on).	1031
DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS	The count of simultaneous requests reached the maximum allowed value. This variable is set in the <code>manager.conf</code> configuration file. The default is 300.	1032
DIVA_ERR_TAPE_DOESNT_EXIST	There is no tape associated with the given barcode.	1033
DIVA_ERR_INVALID_INSTANCE_TYPE	Cannot partially restore this instance.	1034
DIVA_ERR_OBJECT_PARTIALLY_DELETED	The specified object has instances that are partially deleted.	1036
DIVA_ERR_COMPONENT_NOT_FOUND	The specified component (file) is not found.	1038
DIVA_ERR_OBJECT_IS_LOCKED	Attempted to restore an object that has been locked. A locked object cannot be Restored or Copied to New.	1039
DIVA_ALL_REQUESTS	Specify all requests. Used by <code>DIVA_cancelRequest</code> .	-2
DIVA_ALL_INSTANCE	Specify all instances. Used by <code>DIVA_release</code> .	-1
DIVA_ANY_INSTANCE	Allow Manager to choose the instance.	-1

**Table A–2 (Cont.) API Static Constants**

Static Constant Name	Description	Value
DIVA_DEFAULT_REQUEST_PRIORITY	The default request priority. This is used if no specific priority is selected when the request is configured.	-1
DIVA_REQUEST_PRIORITY_MIN	The default minimum request priority.	Default = 0
DIVA_REQUEST_PRIORITY_LOW	The default low request priority.	Default = 25
DIVA_REQUEST_PRIORITY_NORMAL	The default normal request priority.	Default = 50
DIVA_REQUEST_PRIORITY_HIGH	The default high request priority.	Default = 75
DIVA_REQUEST_PRIORITY_MAX	The default maximum request priority.	Default = 100
DIVA_MEDIA_FORMAT_UNKNOWN	The specified tape format is unknown.	-1
DIVA_MEDIA_FORMAT_LEGACY	The specified media format for the group or array is <i>Legacy</i> .	0
DIVA_MEDIA_FORMAT_AXF	The specified media format for the group or array is AXF 0.9.	1
DIVA_MEDIA_FORMAT_AXF_10	The specified media format for the group or array is AXF 1.0.	2
DIVA_OFFSET_BYTE_BEGIN	__int64 - The beginning byte of the file.	0
DIVA_OFFSET_BYTE_END	__int64 - The ending byte of the file.	-1
DIVA_OFFSET_INVALID	__int64 - The specified timecode offset is invalid.	-2
DIVA_OFFSET_TC_BEGIN	string - The file's beginning timecode.	00:00:00:00
DIVA_OFFSET_TC_END	string - The file's ending timecode.	99:99:99:99

---

---

# Glossary

## **Archive Related Operations Initiator**

An entity submitting requests to DIVArchive (typically, an automation process).

## **Array**

In DIVArchive, an array designates a collection of disks identified by their name as they are declared in the DIVArchive configuration. A disk name is associated with a mounting point. Archive requests can be submitted with an array as the destination. DIVArchive is responsible for choosing the disk location to write the data to when several disks belong to the same array.

## **Archive Exchange Format (AXF)**

The Archive Exchange Format (AXF), or AXF Media Format, is based on a file and storage media agnostic encapsulation approach which abstracts the underlying file system, operating system, and storage technology making the format truly open and non-proprietary.

## **Category**

Part of the access key to an object. Categories are an approach to linking the object with the user activity field. It must not be confused with a **Group**, which is a storage concept.

## **Complex Object**

An object is defined as complex when it contains one thousand (this is the default, but the value is configurable) or more components. Complex object handling may differ from non-complex objects as noted throughout this document.

## **Critical Section**

A piece of code that accesses a shared resource (data structure or device) that must not be concurrently accessed by more than one execution thread.

## **Destination**

A system that receives restored data in the DIVArchive system (for example, video servers, remote computers, FTP servers, and so on). Destinations can also be used as a **Source** certain operations.

## **Digital Moving-Picture Exchange (DPX)**

The Digital Moving-Picture Exchange (DPX) format is a high quality video format that consists of one or more files for each frame of video. This format is likely to be used with complex objects.

**Externalization**

An object instance is ejected (externalized) when one of the tapes containing the instance's elements is ejected. An object is ejected when all of its instances are ejected. An object is considered inserted when at least one instance of the object is inserted.

**Group**

A group is a logical notion for characterizing a set of object instances. This concept has a direct influence on the instance's storage policy for tapes. Instances of the same group will be stored on the same tapes. However, objects cannot have multiple instances stored on the same tape.

Groups are based on the DIVArchive Tape Set. Each tape inserted in the system is assigned to a Set. Groups are then associated with a single Set. Multiple groups may be associated with the same set. *No group can use the set number 0.*

Several kinds of tape can be used in a DIVArchive system. Groups can be defined either by using a Set, in which you assign only tapes of the same type, or by defining the Set in which you can mix tape types. Therefore, the first case specifies the tape type that stores the object instance. See [Set \(of Tapes\)](#) later in this section for more information.

**Initiator**

See [Archive Related Operations Initiator](#) previously described.

**Legacy Format**

DIVArchive proprietary storage format used in DIVArchive releases 1.0 through 6.5.

**Media Format**

Tapes and disks may be formatted as either [Archive Exchange Format \(AXF\)](#) or [Legacy Format](#). The format is set for tape groups and disk arrays during configuration.

**Medium (Media)**

A set of storage resources. Currently DIVArchive provides two types of media: Groups of Tapes and Arrays of Disks. The `DIVA_archiveObject()` and `DIVA_copyToGroup()` requests transfer objects to a Medium.

**Migration**

Copying of data from a DIVArchive media to a tape (Archive operation) or from a tape to a DIVArchive media (Restore operation).

**Mutual Exclusion (Mutex)**

Mutual Exclusion (mutex) avoids the simultaneous use of a common resource (that is, mutual exclusion among threads).

**Name**

Part of the access key to an object. Names (file names) typically identify the object based on the content within the object.

**Object**

In DIVArchive objects are archive entries. An object is identified by a pair ([Name](#) and [Category](#)) and contains one or more components. A component is the DIVArchive representation of a file. The components are stored in DIVArchive as an [Object Instance](#). Also see [Complex Object](#).

**Object Instance**

The mapping of an object's components onto a set of storage resources belonging to the same storage space. Deleting instances cannot result in deleting the related object and therefore the deletion of an instance, when that instance is unique, is not permitted.

**Repack**

Elimination of blank blocks between two objects on a tape (these blocks are caused by the deletion of objects), by moving the objects to a different, empty tape.

**Request**

A request is an operation running in DIVArchive which progresses through steps (migration, transfer, and so on) and ends as either **Completed**, **Aborted**, or **Cancelled**.

**Resource**

Used to denote the necessary elements involved for processing requests (for example, Actors, Managers, Disks, Drives, and Tapes).

**Set (of Tapes)**

Every tape in a DIVArchive system belongs to one and only one Set. If the tape is not available to DIVArchive, it belongs to **Set #0**, otherwise it belongs to a set with a strictly positive ID (for example, **Set #1**). Each **Group** is associated with a Set. When the group needs an additional tape, it takes it from its associated Set.

**Source**

A system that produces data to be archived in the DIVArchive system (for example, video servers, browsing servers, remote computers, and so on). Sources can also be used as a **Destination** for certain operations.

**Spanning**

Splitting an object's components onto several tapes (typically two). This can occur when the component size is larger than the remaining size left on the initial tape.

**Transfer**

Copying data from a **Source** to a DIVArchive media (**Archive** operation) or from a DIVArchive media to a **Destination** (**Restore** operation). See **Request** for more information.

**Universally Unique Identifier (UUID)**

A Universally Unique Identifier (UUID) uniquely identifies each object created in DIVArchive across all Oracle customer sites. Objects created using the **Copy As** request are not assigned a UUID. An object created by a **Copy As** request contains the same UUID as that of the source object.

