# Oracle® DIVArchive

C++ API Reference Manual

Release 7.3

**E64034-03**

July 2016

ORACLE®

Oracle DIVArchive C++ API Reference Manual, Release 7.3

E64034-03



Primary Author: Lou Bonaventura

# Table of Contents

# Tables Index

# 1 Introduction

## *1.1 Overview*

This reference manual contains a detailed description of the functionality of the Oracle DIVArchive / Oracle DIVAnet Application Programmer's Interface (*API*).

The DIVArchive API is written in C++. All of the definitions are contained in the include file called **`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. The reader must consider that these qualifiers are equivalent to the following macro definitions:

```
#define IN

#define OUT
```

In this document we use *structure* to identify both C-like structures and classes which have only public data members and no function members[1]. Interfaces described in this document show only data members, not constructors or destructors.

The DIVArchive / DIVAnet API use only *standard data types* provided directly by the C++ language plus the *vector data type* provided by the *Standard Template Library* (*STL*). For more information about the *vector data type* refer to the STL documentation.

**Note: Oracle no longer supports the DIVArchive API under the Solaris Operating System.**

DIVArchive version 7.3 does not currently support the following API calls and features when used with Complex Objects. This means that even if they are enabled, they will not be executed and no warnings will be generated.

- **`VerifyFollowingArchive`**
- **`VerifyFollowingRestore`**
- **`DeleteOnSource`**
- **`DeleteFile`**
- **`getObjectListbyFileName`**
- Copying Complex Objects to Legacy formatted media.
- The **`getObjectInfo`** and **`getObjectDetailsList`** will only return a single file.

[1] Operators *new* and *delete* are not considered function members.

## 1.2 Document Conventions

The following conventions are used with respect to text:

Normal Standard Text.

*Italic* Used to emphasize a term or variable.

**Bold** Used to emphasize critical information.

6.1 Refers to a section or sub-section in the document.

`Courier New` Used for system screen output and system commands.

## 1.3 Definitions, Acronyms, and Special Terms

*Table 1: Definitions, Acronyms, and Special Terms*

| Term | Definition |
|---|---|
| **Administrator** | Person performing DIVArchive operations which are not automatic. |
| **API** | Application Programming Interface |
| **Archive Related Operations Initiator** | An entity submitting requests to DIVArchive (*typically, an automation process*) |
| **Array** | An Array designates a collection of disks designated 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 when several disks belong to the same array. |
| **AXF (*or AXF Media Format*)** | The Archive Exchange Format (*AXF*) 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 (*see Object*). Categories are an approach to linking the object with the user activity field. It must not be confused with the *Group* idea, which is a storage concept. |
| **Complex Object** | An Object (*see below*) is defined as a Complex Object when it contains 1,000 (*configurable*) or more components. Complex object handling may differ from non-complex objects as noted throughout this document. |

| Term | Definition |
|---|---|
| **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 on which archived objects are restored. |
| **DPX** | Digital Moving-Picture Exchange format. This 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 when one of the tapes containing the Object's Instance 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 idea has a direct influence on the instance's storage policy on tapes. Instances of the same group will be stored on the same tapes. However, objects cannot have multiple instances stored on the same tape.<br><br>The Group concept is based on the DIVArchive 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. **Caution: No group can use the set number 0**.<br><br>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 mix tape types. The first case therefore specifies the tape type that is used to store the Object Instance. |
| **Initiator** | Refer to Archive Related Operations Initiator. |
| **Legacy Format** | DIVArchive proprietary storage format used in version 1.0 through 6.5.1. |
| **Media Format** | Tapes and Disks may be formatted as either AXF or Legacy (*format used prior to version 7.0*). The format is set for Tape Groups and Disk Arrays during configuration. |
| **Medium** | Set of storage resources. Currently DIVArchive provides two types of media: Groups of Tapes and Arrays of Disks. **`DIVA_archiveObject()`** and **`DIVA_copyToGroup()`** requests transfer to a Medium (*Media*). |

| Term | Definition |
|---|---|
| **Migration** | Copying of data from a DIVArchive Media to a tape (*Archive Operation*) or from a tape to a DIVArchive Media (*Restore Operation*). |
| **Mutex** | Mutual Exclusion (*mutex*) avoids the simultaneous use of a common resource (*i.e.: mutual exclusion amongst threads*). |
| **Object** | Objects are archive entries. An object is identified by a pair (*Name, Category*) and contains Components**.** A Component is the DIVArchive representation of a file. The Components are stored in DIVArchive as Object Instances. Also see Complex Object above. |
| **Object Instance** | 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 though steps (*migration, transfer etc.*) and ends as *Completed*, *Aborted,* or *Cancelled*. |
| **Resource** | Used to denote the necessary elements involved for processing requests (*e.g. Oracle DIVArchive Actors, Disk, Drive, and Tape*). |
| **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 (*e.g.: 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 (*e.g.: video servers, browsing servers, remote computers, etc.*). |
| **Spanning** | Splitting an Object's Component onto several tapes (*usually two*); this may occur when the component size is larger than the remaining size left on the initial tape. |
| **STL** | Standard Template Library |

| Term | Definition |
| --- | --- |
| **Transfer** | Copying data from a Source to a DIVArchive Media (*Archive Operation*) or from a DIVArchive Media to a Destination (*Restore Operation*). |
| **UUID** | Universally Unique Identifier to uniquely identify each object created in DIVArchive across all Oracle customer sites, except for objects created via **Copy As Requests**. An object created via a **Copy As Request** will contain the same UUID as that of the source object. |

# 2 DIVArchive API Usage and Operations

## 2.1 Visual C++ Compiler on Windows

### 2.1.1 Supported Platforms

The DIVArchive API for Windows is supported on the following Operating Systems:

- Microsoft Windows XP
- Microsoft Windows 2000
- Microsoft Windows 2003
- Microsoft Windows Server 2008
- Microsoft Windows Server 2008 x64
- Microsoft Windows Server 2008 R2

There are two separate versions of the DIVArchive API for Windows; 32-bit and 64-bit. While the 32-bit variant can be used on both x86 and x64 platforms, the 64-bit variant requires 64-bit platform.

### 2.1.2 Supported Compilers

The DIVArchive API has been compiled and tested using the following compilers:

- Microsoft Visual C++ 6.0
  - Including Microsoft Platform SDK (*February 2003*)
- Microsoft Visual C++ .NET (*Version 7*)
  - Including Microsoft Platform SDK for Windows Server 2003 R2 (*March 2006*)
- Microsoft Visual C++ 2005 (*Version 8*)
  - Including Microsoft Platform SDK for Windows Server 2003 R2 (*March 2006*)
- Microsoft Visual C++ 2008 (*Version 9*)
  - Including Microsoft Platform SDK 6.0a (*November 2007*)

### 2.1.3 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/or Unicode compatibility. The different options may be found in the following folders:

*Table 2: API Library Options*

| Build Directory | Library Description |
|---|---|
| `Static_Release` | Static Library |
| `Static_Release_Unicode` | Static Library, Unicode compatible |
| `Dynamic_Release` | Dynamic Library |
| `Dynamic_Release_Unicode` | Dynamic Library, Unicode compatible |
| `Static_Debug` | Static Library with debug support. |
| `Static_Debug_Unicode` | Static Library with debug support and Unicode compatible. |
| `Dynamic_Debug` | Dynamic Library with debug support. |
| `Dynamic_Debug_Unicode` | Dynamic Library with debug support and Unicode compatible. |

### 2.1.4 Compilation

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

The following table shows the library path that corresponds to each runtime library. The runtime library is normally changed automatically, depending upon the selected build configuration.

*Table 3: Compiler Library Paths*

| Runtime Library Option | Library File Directory |
|---|---|
| Multithreaded | **`Static_Release`** or **`Static_Release_Unicode`** |
| Debug Multithreaded | **`Static_Debug`** or **`Static_Debug_Unicode`** |
| Multithreaded DLL | **`Dynamic_Release`** or **`Dynamic_Release_Unicode`** |
| Debug Multithreaded DLL | **`Dynamic_Debug`** or **`Dynamic_Debug_Unicode`** |

- Include **`DIVArchive API.lib`** or the path to this file in the link settings (*See Initiator examples*).
- The API may be included in an application compiled either with the IDE or a script using the command line compiler.
- Once your application has been 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.

### 2.1.5 Sample API Usage

The Initiator program 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. Initiator is included with the DIVArchive API. Use the following project files to view the compiler settings and build the program:

*Table 4: Sample API Usage*

| Compiler IDE | Project File |
|---|---|
| Visual C++ 6.0 | **API\docs\samples\initiator.dsp** |
| Visual C++ .NET (*Version 7*) | **API\docs\samples\initiatorVc7.vcproj** |
| Visual C++ 2005 (*Version 8*) | **API\docs\samples\initiatorVc8.vcproj** |
| Visual C++ 2008 (*Version 9*) | **API\docs\samples\initiatorVc9.vcproj**<br>**API\docs\samples\initiator64Vc9.vcproj (64-bit API)** |

## 2.2 GCC C++ Compiler on Linux

### 2.2.1 Supported Platforms

The DIVArchive API for Linux is supported on the following operating systems:

- SuSe 9.0 (*x86*)
- Fedora Core Release 5 (*x86*)

The API was built with the GCC C++ compiler and **`glibc`** library installed by default:

*Table 5: Linux API Libraries*

| Operating System | Linux Kernel | GCC | glibc Library |
|---|---|---|---|
| Suse 9.0 | 2.4.21 | 3.3.1 | 2.3.2 |
| Fedora Core 5 | 2.6.20 | 4.1.1 | 2.4 |

Use the **`uname`** command to get the Linux kernel version.

**Example:**

```
$ uname –r
2.6.20-1.2300.fc5smp
```

The following command returns the GCC version:

```
$ gcc –-version
gcc (GCC) 4.1.1 20070105 (Red Hat 4.1.1-51)
```

The DIVArchive API may work on other Linux platforms not listed in this document, but is officially validated only in the environments described here.

While both versions of the DIVArchive API are currently supported, the older version built on SuSe Linux 9.0 is only provided upon request. For all development projects, use of the newer version is **strongly** recommended. Support of SuSe Linux 9.0 will be discontinued starting with the next major release of DIVArchive.

### 2.2.2 Compilation

Two variants of the DIVArchive API shared library **`libDIVArchive API.so`** are delivered for Linux platforms. The non-Unicode version is located in the **`DIVA/api/lib`** folder and the Unicode-aware version is located in the **`DIVA/api/lib_Unicode`** folder. Both variants are built in **Release** mode and do **not** contain symbolic information.

Any header files required to compile an application with DIVArchive API libraries are delivered in the **`DIVA/api/include`** folder.

Refer to the sample **`makefiles`** provided in the **`DIVA/api/doc/samples`** folder for platform specific compiler and linker options.

### 2.2.3 Samples

For reference, a sample application is provided in the **`DIVA/api/doc/samples`** folder along with its source code. The Visual Studio project file for Microsoft Windows, and sample **`makefiles`** for Linux platforms are also provided.

To build the reference application for a specific platform, use the corresponding **`makefile`**. For example, to build a Unicode-aware variant for Linux, run the following command:

```
$ make –f Makefile.linuxUnicode
```

## 2.3 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 so, while 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 critical sections 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()`**.

## 2.4 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 30 connections and includes GUI connections and all API connections. Once the configured limit is reached, the API will not allow additional connections to be created. Please refer to the **`manager.conf`** file for additional information.

**Note: 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.**

## 2.5 Using Unicode Strings in the DIVArchive API

The DIVArchive API (*and other DIVArchive components*) support wide character strings. To be able to use **`wchar_t`** and **`wstring`**, you must define the **`_UNICODE`** constant before including the **`DIVAapi.h`** header file.

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

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 6: Unicode Strings*

| Type | **`_UNICODE`** Not Defined | **`_UNICODE`** Defined |
|---|---|---|
| **`DIVA_STRING`** | **`string`** | **`Wstring`** |
| **`DIVA_CHAR`** | **`char`** | **`wchar_t`** |

## 2.6 DIVArchive Version Compatibility

The DIVArchive and DIVAnet Systems are backward compatible with all previous and current versions of the DIVArchive C++ API. Therefore, DIVArchive C++ API version 6.5.x.x.x would be compatible with any DIVArchive version 6.5, 7.0, and later.

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

## 2.7 Alternate APIs

The API described by this document is for use with applications implemented in C++. Additional APIs are available:

- **DIVArchive Java API** – A set of libraries, samples and documentation for usage with applications implemented in Java. Please see the DIVArchive Java API documentation for more information.
- **DIVArchive Web Services API** – A set of interface definition files and documentation for universal usage by applications supporting Web Services communications. Please see the DIVArchive Web Services API documentation for more information.

## 2.8 Session Management Commands

### 2.8.1 DIVA_getApiVersion

#### Synopsis

```
#include “DIVAapi.h”

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

| Variable | Description |
|---|---|
| **version** | Points to a string that contains the major part of the release for this API. |

#### Description

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

### 2.8.2 DIVA_connect

#### 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
);
```

| Variable | Description |
|---|---|
| **managerAddress** | IP Address of the DIVArchive Manager. |
| **portNumber** | Port on which the DIVArchive Manager is listening. The default port is pointed to by the constant value **DIVA_MGER_DEFAULT_PORT.** |
| **userName** | User Name. |
| **password** | User Password. |
| **applicationName** | Application Name. |
| **userInfo** | User specific and supplied information. |

### Description

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()**.

### 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 this thread exits the **DIVA_connect()** function before being able to open or close the connection.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been 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 has been broken. |

| Value | Description |
|---|---|
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |
| **DIVA_ERR_NO_ARCHIVE_SYSTEM** | Problem when establishing a connection with the specified DIVArchive System. |
| **DIVA_ERR_WRONG_VERSION** | Release version of the API and the Manager are not compatible. |
| **DIVA_ERR_ALREADY_CONNECTED** | A connection is already open. |

**See Also:** **DIVA_disconnect**

### 2.8.3 DIVA_disconnect

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_disconnect ()
```

#### Description

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**.

#### 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 this thread exits the **DIVA_disconnect()** function before being able to open or close the connection.

#### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_BROKEN_CONNECTION** | The connection with the DIVArchive Manager has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |
| **DIVA_ERR_DISCONNECTING** | Problem when disconnecting. The connection is still considered to be open. |

**See Also:** **DIVA_connect**

## 2.9 Requests and Commands

### 2.9.1 DIVA_addGroup

#### 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
);
```

| Variable | Description |
|---|---|
| **groupName** | Name of the Group to be added. |
| **associatedSet** | DIVArchive Set of Tapes to associate with the new group. This value must be strictly greater than 0. |
| **comment** | Text describing 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** | Number of tapes reserved for Worst Fit Repacking. |
| **mediaFormatId** | Data format to be used by the tapes assigned to this group. (***DIVA_MEDIA_FORMAT_LEGACY, DIVA_MEDIA_FORMAT_AXF,*** *or* ***DIVA_MEDIA_FORMAT_AXF_10***). Refer to Table 1: Definitions, Acronyms, and Special Terms for more format information. |

#### Description

Adds a new group.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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_GROUP_ALREADY_EXISTS** | The specified group already exists. |

### 2.9.2 DIVA_archiveObject

#### 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              filesPathRoot,
IN vector<DIVA_STRING>      filenamesList,
IN DIVA_ARCHIVE_QOS         qualityOfService,
IN int                      priorityLevel,
IN DIVA_STRING              comments,
IN DIVA_STRING              archiveOptions,
OUT int                     *requestNumber
);
```

| Variable | Description |
|---|---|
| **objectName** | Name of the Object to be archived. |
| **objectCategory** | Category of the Object. |
| **source** | Name of the Source (*e.g. video server, browsing server*). This name must be known to the DIVArchive Configuration Description. |

| Variable | Description |
|---|---|
| **mediaName** | The Tape Group or Disk Array on which the object is to be saved. The media may be defined as follows:<br>1. **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*).<br>2. **SP Name** – Provide a Storage Plan Name as defined in the configuration. The object will be saved to the default media specified in the SP and assigned to the specified SP.<br>3. **Both 1 and 2: Name "&" SP Name** – The object is saved to the specified media as in number 1 above. The object is assigned to the specified SP as in number 2 above. The Media Name and the SP Name must be separated by the delimiter "&" (*configurable*).<br>When this parameter is a null string, the default group of tapes called **DEFAULT** is used.<br>Complex Objects can only be saved to AXF media types. |
| **filesPathRoot** | Root folder for the files specified by the **filenamesList** parameter. |
| **filenamesList** | List of file pathnames relative to the folder specified by the **filesPathRoot** parameter. When **filesPathRoot** is null, pathnames must be absolute names.<br>For DIVArchive version 7.1.2 and later ONLY:<br>If the **-gcinfilelist** option is specified, the Genuine Checksum is included with a colon separator between the file name and the GC value.<br>**test1.txt:a6f62b73f5a9bf380d32f062f2d71cbc**<br>**test2.txt:96bf41e4600666ff69fc908575c0319c** |

| Variable | Description |
|---|---|
| **qualityOfService** | One of the following codes:<br>**DIVA_QOS_DEFAULT:** Archiving is performed according to the default Quality Of Service (*currently: direct and cache for archive operations*).<br>**DIVA_QOS_CACHE_ONLY:** Use cache archive only.<br>**DIVA_QOS_DIRECT_ONLY:** Use direct archive only. No Disk Instance is created.<br>**DIVA_QOS_CACHE_AND_DIRECT:** Use cache archive if available or direct archive if cache archive is not available.<br>**DIVA_QOS_DIRECT_AND_CACHE:** Use direct archive if available or cache archive if direct archive is not available.<br>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:<br>**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** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY.** With this value the Manager uses the default priority for this request (*default request priority is defined in the Manager configuration*).<br>Using another value (*out of the range [0...100] or predefined values*) yields a **DIVA_ERR_INVALID_PARAMETER** error. |
| **comments** | Optional information describing the object (*can be a null string*). |

| Variable | Description |
|---|---|
| **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:<br>• A null string to specify no options.<br>• **-r**: Specifies that every name in **filenamesList** that refers to a folder must be scanned recursively. This also applies when a *Files Path Root* is specified and '*****' is used to designate the file(s) to be archived. This option may be used when archiving from a local source or from a standard FTP Server.<br>• **-login:** Login is used for some sources. This option obsoletes the –gateway option from the previous version.<br>• **-pass:** Password used in conjunction with the **-login** option for some sources.<br>For DIVArchive version 7.1.2 and later ONLY:<br>**-gcinfilelist [gcType]**: Specifies that Genuine Checksum (*GC*) values are included in the file names list. The value of **gcType** must match the Manager Default Checksum Type as specified in the DIVArchive Configuration (*this is MD5 by default*). The GC values are then used to verify the transfer from the Source. |
| **requestNumber** | Request Number assigned to this request. This number is used for querying the status or cancelling this request. |

## Description

Submits an **Object Archive Request** to the DIVArchive Manager. This function returns as soon as the Manager accepts the request.

To check that the operation completes successfully, the application must call the function **DIVA_getRequestInfo().**

## Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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** | Count of simultaneous requests reached the maximum allowed value. This variable is set in the **manager.conf** configuration file. The default value is 300. |
| **DIVA_ERR_OBJECT_ALREADY_EXISTS** | An object with the name and category already exists in the DIVArchive System. |
| **DIVA_ERR_GROUP_DOESNT_EXIST** | The Group of Tapes or the Array of Disks does not exist. |
| **DIVA_ERR_SOURCE** or **DESTINATION_DOESNT_EXIST** | The specified Source/Destination is not known by the DIVArchive System. |

### 2.9.3 DIVA_associativeCopy

#### 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
);
```

| Variable | Description |
|---|---|
| **objectsInfo** | Pointer to a list of objects defined by a pair (*Name, Category*). |
| **groupName** | Name of the group where the new instance will be located.<br>**Note: Associative Copy to Disk Array is not available.**<br>Complex Objects may be saved only to AXF media types. |
| **priorityLevel** | Level of priority for this Request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY.** With this value, the Manager uses the default priority for this request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] or predefined values*) yields a **DIVA_ERR_INVALID_PARAMETER** error. |
| **options** | Optional string attribute for specifying additional parameters to the request. |
| **requestNumber** | Number identifying the request. |

### Description

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 has been copied onto 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.
- 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. Request aborts (*and is retried once*) instead of spanning. If the sum of the size of the objects to copy exceeds the capacity of every individual tape present in the library, the request aborts.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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. |

| Value | Description |
|---|---|
| **DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS** | 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_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 Oracle DIVArchive Actor could provide a Disk Instance. |
| **DIVA_ERR_GROUP_DOESNT_EXIST** | The Group does not exist. |
| **DIVA_ERR_OBJECT_IN_USE** | The Object is currently in use (being Archived, Restored, Deleted, etc.). |
| **DIVA_ERR_OBJECT_PARTIALLY_DELETED** | The specified object has instances that are partially deleted. |

**See Also:**

- **DIVA_archiveObject**
- **DIVA_copyToGroup**

### 2.9.4 DIVA_cancelRequest

#### Synopsis

```
#include "DIVAapi.h"

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

| Variable | Description |
|---|---|
| **requestNumber** | Number identifying the request to be cancelled. This parameter can be set to **DIVA_ALL_REQUESTS** to cancel all cancellable requests. |
| **options** | Optional string attribute for specifying additional parameters to the request. |

#### Description

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

#### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |

| Value | Description |
|---|---|
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |
| **DIVA_ERR_NO_SUCH_REQUEST** | **requestNumber** identifies no request. |

**See Also:** **DIVA_getRequestInfo**

### 2.9.5 DIVA_changeRequestPriority

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_changeRequestPriority (
IN int          requestNumber,
IN int          priorityLevel,
IN DIVA_STRING  passThruOptions
);
```

| Value | Description |
|---|---|
| **requestNumber** | Number identifying the request to be changed. |
| **priorityLevel** | The **priorityLevel** can be in the range [0...100]. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>The use of **DIVA_DEFAULT_REQUEST_PRIORITY** is **not** allowed with this function.<br>Using another value (*out of the range [0...100] or predefined values*) yields a **DIVA_ERR_INVALID_PARAMETER** error. |
| **passThruOptions** | Optional string attribute for specifying additional parameters to the request. |

#### Description

Submits a **Change Request Priority Request** to the DIVArchive Manager. 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()**.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |
| **DIVA_ERR_NO_SUCH_REQUEST** | **requestNumber** identifies no request. |
| **DIVA_ERR_INVALID_PARAMETER** | A parameter value has not been understood by the DIVArchive Manager. |

**See Also:** **DIVA_getRequestInfo**

### 2.9.6 DIVA_copyToGroup or DIVA_copy

#### Synopsis

```
#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
);
```

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

| Variable | Description |
|---|---|
| **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 (*this may result in an error if several objects have the same name*). |
| **instanceID** | Instance's identifier. **DIVA_ANY_INSTANCE** as the Instance ID means that DIVArchive will choose the appropriate instance. |
| **mediaName** | Media (*Tape Group or Disk Array*) on which the new instance will be located. |

| Variable | Description |
|---|---|
| **priorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY.** With this value, the Manager uses the default priority for this request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] or predefined values*) yields a **DIVA_ERR_INVALID_PARAMETER** error. |
| **options** | Optional string attribute for specifying additional parameters to the request. |
| **requestNumber** | Number identifying the request. |

## Description

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. To check that the operation was successful the application must call the function **DIVA_getRequestInfo()**.

In the event 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.

**Note: A Tape Group may contain two instances of the same object. In this case, DIVArchive will abort the request if both instances cannot be written on two different tapes. Similarly, a Disk Array can contain two instances of the same object, but DIVArchive will abort the request if the new instance can't be written on a different disk. In conclusion, there can be a maximum of only one instance of each object per disk or tape.**

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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** | 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_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. |

| Value | Description |
|---|---|
| **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** | 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 Group does not exist. |
| **DIVA_ERR_OBJECT_IN_USE** | The Object is currently in use (being Archived, Restored, Deleted, etc.). |
| **DIVA_ERR_OBJECT_PARTIALLY_DELETED** | The specified object has instances that are partially deleted. |

**See Also:** **DIVA_archiveObject**

### 2.9.7 DIVA_copyToNewObject

#### Synopsis

```
#include "DIVAapi.h"

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

| Value | Description |
|---|---|
| **source** | Description of the Object or Object Instance to be copied:<br>**source.objectName** — Source Object Name (*required*).<br>**source.objectCategory** — Source Object Category (*required*).<br>**source.group** — Group/Array of the Source Object Instance (*optional*). If specified, DIVArchive will use that instance as a source.<br>**source.instanceID** — **InstanceID** of the Source Object Instance (*optional*). If specified (*not equal to* ***DIVA_ANY_INSTANCE***), DIVArchive will use that instance as a Source. **source.group** will be ignored if **instanceID** is specified.<br>If both **source.group** and **source.instanceID** are omitted, DIVArchive will use the most suitable instance (*providing the best performance*) as a source. |

| Value | Description |
|---|---|
| **target** | Description of the Target Object:<br>**target.objectName** — Target Object Name (*required*).<br>**target.objectCategory** — Target Object Category (*required*).<br>**target.group** — See below.<br>**target.instanceID** — Ignored.<br>Either Object Name or Category (*or both*) must be different from Name/Category of the Source Object.<br>Request will fail if the Target Object already exists in DIVArchive. |
| **attrs** | Request attributes:<br>**attrs.priority** — Request Priority (*optional*), default priority used if not set explicitly. Possible values: 0 (*lowest*) – 100 (*highest*).<br>**attrs.qos** — Ignored, Quality Of Service is not applicable to this request.<br>**attrs.comments** — Target Object Comments (*optional*). If not set, comments from the Source Object are inherited.<br>**attrs.options** — Ignored; request has no additional options. |
| **requestNumber** | Number identifying the request (*returned by DIVArchive*). |

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

| Value | Description |
|---|---|
| **objectName** | Source Object Name. |
| **objectCategory** | Source Object Category. |
| **objectMedia** | Group/Array of the Source Object Instance (*optional*). If specified (*not empty*), DIVArchive will use that instance as a source.<br>Complex Objects may be saved only to AXF media types. |
| **objectInstanceID** | **InstanceID** of the Source Object Instance (*optional*). If specified (*not equal to* ***DIVA_ANY_INSTANCE***), DIVArchive will use that instance as a source. **objectMedia** will be ignored if **instanceID** is specified.<br>If both **objectMedia** and **instanceID** were not specified, DIVArchive will use the most suitable instance (*providing the best performance*) as a source. |
| **newObjectName** | Target Object Name. |

| Value | Description |
|---|---|
| **newObjectCategory** | Target Object Category.<br>Either Object Name or Category (*or both*) must be different from Name/Category of the Source Object.<br>Request will fail if the Target Object already exists in DIVArchive. |
| **newObjectInstanceMedia** | The Tape Group or Disk Array on which the object is to be saved. The media may be defined as follows:<br>1. **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*).<br>2. **SP Name** – Provide a Storage Plan Name as defined in the configuration. The object will be saved to the default media specified in the SP and assigned to the specified SP.<br>3. **Both 1 and 2: Name "&" SP Name** – The object is saved to the specified media as in number 1 above. The object is assigned to the specified SP as in number 2 above. The Media Name and the SP Name must be separated by the delimiter "&" (*configurable*). |
| **comments** | Target Object Comments (*optional*). If empty, comments from the Source Object are used. |
| **priorityLevel** | Request Priority. Possible values: 0 (*lowest*) – 100 (*highest*), and **DIVA_DEFAULT_REQUEST_PRIORITY** (*use default Request Priority*). |
| **options** | Optional string attribute for specifying additional parameters to the request. |
| **requestNumber** | Number identifying the request (*returned by DIVArchive*). |

## Description

Submits a request for copying an archived object to a new object, with another name and/or category, to the DIVArchive Manager and the Manager chooses the appropriate instance as the source of the copy. 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()`**.

In the event 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.

All types of transfers (*Disk to Disk, Disk to Tape, Tape to Disk, and Tape to Tape*) are supported.

## Return Values

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

| Value | Description |
|---|---|
| **`DIVA_OK`** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **`DIVA_ERR_NOT_CONNECTED`** | No open connection. |
| **`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 has been broken. |
| **`DIVA_ERR_TIMEOUT`** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **`DIVA_API_TIMEOUT`** variable and equals 180 seconds by default. |
| **`DIVA_ERR_UNKNOWN`** | An unknown status has been received from the DIVArchive Manager. |
| **`DIVA_ERR_INTERNAL`** | An internal error has been 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. |

| Value | Description |
|---|---|
| **DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS** | 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_INSTANCE_DOESNT_EXIST** | Instance specified for restoring this object does not exist. |
| **DIVA_ERR_OBJECT_ALREADY_EXISTS** | Target Object already exists in DIVArchive. |
| **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** | 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 Group does not exist. |
| **DIVA_ERR_OBJECT_IN_USE** | The Object is currently in use (being Archived, Restored, Deleted, etc.). |
| **DIVA_ERR_OBJECT_PARTIALLY_DELETED** | The specified object has instances that are partially deleted. |

**Notes:**

- Available since DIVArchive v5.7.
- **See Also:** **DIVA_copyToGroup**

### 2.9.8 DIVA_deleteGroup

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_deleteGroup (
IN DIVA_STRING        groupName
);
```

| Variable | Description |
|---|---|
| **groupName** | Name of the Group to be deleted. |

#### Description

Deletes the group passed as an argument. Deleting a group is only possible when the group is empty.

#### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

| Value | Description |
| --- | --- |
| **DIVA_ERR_INVALID_PARAMETER** | A parameter value has not been understood by the DIVArchive Manager. |
| **DIVA_ERR_GROUP_DOESNT_EXIST** | The Group does not exist. |
| **DIVA_ERR_GROUP_IN_USE** | The Group contains at least one Object Instance. |

### 2.9.9 DIVA_deleteInstance

#### 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 int                instanceID,
IN DIVA_STRING        mediaName,
IN int                priorityLevel,
IN DIVA_STRING        options,
OUT int               *requestNumber
);
```

| Variable | Description |
|---|---|
| **objectName** | Name of the Object to be deleted. |
| **objectCategory** | Category assigned to the object when it was archived. This parameter can be a null string (*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 will be deleted from the media. |

| Variable | Description |
|---|---|
| **priorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY.** With this value the Manager uses the default priority for this request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] or predefined values*) yields a **DIVA_ERR_INVALID_PARAMETER** error. |
| **options** | Optional string attribute for specifying additional parameters to the request. |
| **requestNumber** | Number identifying the request. |

### Description

Deletes an Object Instance.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |

| Value | Description |
|---|---|
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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** | 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_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, etc.). |
| **DIVA_ERR_OBJECT_PARTIALLY_DELETED** | The specified object has instances that are partially deleted. |

**See Also:** **DIVA_getObjectInfo**

### 2.9.10 DIVA_deleteObject

#### 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
);
```

| Value | Description |
|---|---|
| **objectName** | Name of the Object to be deleted. |
| **objectCategory** | Category assigned to the object when it was archived. This parameter can be a null string (*this may result in an error if several objects have the same name*). |
| **priorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY.** With this value the Manager uses the default priority for this request (*default request priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] or predefined values*) yields a **DIVA_ERR_INVALID_PARAMETER** error. |
| **options** | Optional string attribute for specifying additional parameters to the request. |
| **requestNumber** | Request number assigned to this request. This number is used for querying the status or cancelling this request. |

### Description

Submits an **Object Delete Request** to the DIVArchive Manager. The DIVArchive 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()`**.

### Return Values

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

| Value | Description |
|---|---|
| **`DIVA_OK`** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **`DIVA_ERR_NOT_CONNECTED`** | No open connection. |
| **`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 has been broken. |
| **`DIVA_ERR_TIMEOUT`** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **`DIVA_API_TIMEOUT`** variable and equals 180 seconds by default. |
| **`DIVA_ERR_UNKNOWN`** | An unknown status has been received from the DIVArchive Manager. |
| **`DIVA_ERR_INTERNAL`** | An internal error has been 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`** | 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 and is **not** being archived. |
| **`DIVA_ERR_SEVERAL_OBJECTS`** | More than one object with the specified name exists in the DIVArchive Database. |

| Value | Description |
| --- | --- |
| **DIVA_ERR_OBJECT_IN_USE** | The object is currently being read or deleted. |
| **DIVE_ERR_OBJECT_BEING_ARCHIVED** | The specified object does not exist in the DIVArchive database, but it is currently being archived. |

**See Also:**

- **DIVA_getRequestInfo**
- **DIVA_deleteInstance**

### 2.9.11 DIVA_ejectTape

#### 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
);
```

| Variable | Description |
|---|---|
| **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** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY.** With this value the Manager uses the default priority for this request (*default Request Priority is defined in the Manager configuration*).<br>Using another value (*out of the range [0...100] or predefined values*) yields a **DIVA_ERR_INVALID_PARAMETER** error. |
| **requestNumber** | Number identifying the request. |

### Description

Submits an **Eject Request** to DIVArchive. This 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 is currently in use by another request, the **`DIVA_ERR_INVALID_PARAMETER`** status code is returned and no tapes are ejected.

### Return Values

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

| Value | Description |
|---|---|
| **`DIVA_OK`** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **`DIVA_ERR_NOT_CONNECTED`** | No open connection. |
| **`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 has been broken. |
| **`DIVA_ERR_TIMEOUT`** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **`DIVA_API_TIMEOUT`** variable and equals 180 seconds by default. |
| **`DIVA_ERR_UNKNOWN`** | An unknown status has been received from the DIVArchive Manager. |
| **`DIVA_ERR_INTERNAL`** | An internal error has been 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, or at least one of the barcodes refers to a bad tape (*i.e.: unknown tape, offline tape, or tape in use*). |
| **`DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS`** | Count of simultaneous requests reached the maximum allowed value. This variable is set in the **`manager.conf`** configuration file. The default is 300. |

**See Also:** **`DIVA_insertTape`**

### 2.9.12 DIVA_enable_Automatic_Repack

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_enableAutomaticRepack (
IN bool    enable
);
```

| Variable | Description |
|---|---|
| **enable** | True to enable Automatic Repack Scheduling, false to disable. |

#### Description

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 may 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 may be cancelled (*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 (*from this API or from the Control GUI*).

#### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |

| Value | Description |
|---|---|
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

### *2.9.13 DIVA_getArchiveSystemInfo*

#### Synopsis

```
#include "DIVAapi.h"

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

| Variable | Description |
|---|---|
| **info** | Pointer to a **DIVA_GENERAL_INFO** structure that will be modified to include information about the DIVArchive Archive 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
};
```

| Parameter | Definition |
|---|---|
| **actorName** | Name of the DIVArchive Actor. |
| **actorAddress** | The DIVArchive Actor IP Address. |
| **actorIsAvailable** | Determines if the Actor is available. |
| **connectedDrives** | Identifies the connected drives. |
| **repackEnabled** | **True** if tape repack is enabled. |
| **classicEnabled** | Kept for compatibility. This is only **True** only if all seven standard operations are enabled. |
| **cacheArchiveEnabled** | **True** if Cached Archive is enabled. |
| **directArchiveEnabled** | **True** if Direct Archive is enabled. |
| **cacheRestoreEnabled** | **True** if Cached Restore is enabled. |

| Parameter | Definition |
|---|---|
| **directRestoreEnabled** | **True** if Direct Restore is enabled. |
| **deleteEnabled** | **True** if Delete is enabled. |
| **copyToGroupEnabled** | **True** if Copy To Group is enabled. |
| **associativeCopyEnabled** | **True** if Associative Copy is enabled. |
| **cacheForRepack** | **True** if Cached Repack is enabled. |
| **lsmName** | User-friendly Library Storage Module Name. |
| **lsmID** | LSM Unique ID. |
| **lsmIsAvailable** | **True** if this LSM is available for DIVArchive. |
| **driveName** | Drive Name. |
| **driveTypeID** | Drive Type ID. |
| **driveType** | Drive Type Name. |
| **lsmID** | ID of the LSM containing the Drive, see LSM List. |
| **driveIsAvailable** | **True** if Drive is available for DIVArchive. |
| **status** | Status of DIVArchive. |
| **lib_status** | **OK** if at least one ACS is online. See LSM List for details. |
| **totalNumberOfObjects** | Number of Objects managed by this DIVArchive System. |
| **actorsDrivesList** | **<DIVA_ACTOR_AND_DRIVES_DESC>** |
| **lsmList** | **<DIVA_LSM_DESC>** |
| **drivesList** | **<DIVA_DRIVE_DESC>** |
| **numberOfBlankTapes** | Number of blank Tapes that are in a Set associated with at least one Group. Tape(s) may be externalized or write disabled. |
| **remainSizeOnTapes** | In Gigabytes = The sum of remaining size of Tapes 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 permenent storage. (*Only Disks that are currently visible are used in the calculation*). |

| Parameter | Definition |
|---|---|
| **totalSizeOnTapes** | In Gigabytes = The sum of the total size of all Tapes available for DIVArchive (*i.e. in a Set associated with at least one Group*) and of the total size of all Disks accepting storage (*known only if Disk is currently visible*). |
| **capSize** | Number of slots in the default CAP. |
| **pendingRequests** | Number of Pending Requests. |
| **currentRequests** | Number of Current Requests. |
| **numOfAvailableActors** | The number of Actors currently running. |
| **numOfAvailableDrives** | The number of Drives currently **ONLINE**. |
| **numOfAvailableDisks** | The number of Disks currently **ONLINE**. |
| **siteName** | Name of the main site as entered in the Configuration Utility. |
| **siteIpAddress** | Manager IP Address. |
| **sitePort** | Manager port number. |
| **firstUsedRequestId** | First request ID used by the current Manager session. This is **-1** if no requests have been processed. |
| **lastUsedRequestId** | Last request ID used by the current Manager session. This is **-1** if no requests have been processed. |

### Description

Retrieves general information about the DIVArchive System.

**Note: A DIVArchive System communicates with a Robotic System composed of one or more independent ACS (*Automated Cartridge Systems*). The ACS is composed of one or more LSM (*Library Storage Modules*) that can exchange tapes through a PTP (*Pass Through Port*). Each drive is located in a LSM.**

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |

| Value | Description |
|---|---|
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

### 2.9.14 DIVA_getArrayList

#### Synopsis

```
#include "DIVAapi.h"

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

| Variable | Description |
|---|---|
| arraysInfo | 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;
};
```

| Parameter | Definition |
|---|---|
| **arrayDesc** | Array Description. |
| **arrayName** | Array Name. |
| **numberOfDisk** | Number of Disks in the Array. |
| **mediaFormatId** | The format of the data on disks in this array (***DIVA_MEDIA_FORMAT_LEGACY, DIVA_MEDIA_FORMAT_AXF,*** *or* ***DIVA_MEDIA_FORMAT_AXF_10***). Refer to Table 1: Definitions, Acronyms, and Special Terms for more format information. |
| **arrayDiskList** | List of Disks in the 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** | Disk current remaining size. |
| **disk_isWritable** | Flag to check if Disk is writable or not. |
| **disk_maxThroughput** | Maximum throughput of the Disk. |
| **disk_minFreeSpace** | Minimum free space available on Disk. |
| **disk_name** | Name of the Disk. |
| **disk_site** | Name of the Disk Site. |
| **disk_status** | Disk Status. |
| **disk_total_size** | Disk Total Size. |

| Parameter | Definition |
|---|---|
| **disk_array_name** | Name of the Array. |

### Description

The purpose of this function is to provide a list of arrays and disks associated with the arrays in the DIVArchive System. It will also return arrays which don't have any disks associated with them.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **DIVA_ERR_BROKEN_CONNECTION** | The connection with the DIVArchive Manager has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API |

### 2.9.15 DIVA_getFinishedRequestList

#### Synopsis

```
#include "DIVAapi.h"

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

| Variable | Description |
|---|---|
| **batchSize** | The maximum size of the returned list of objects. Must be set to a value no greater than 1000; the recommended setting is 500.<br>Note: 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 (*3 days*).<br>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** | Pointer to the returned data. See the description of **DIVA_FINISHED_REQUEST_INFO** below. It is the responsibility of the user to allocate and delete instances of this class. |

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

| Variable | Description |
|---|---|
| **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** | Pointer to the returned data. See the description of **DIVA_REQUEST_INFO** under the description of **DIVA_getRequestInfo**. |

### Description

Get all of the requests which have finished, starting from the specified number of seconds before the present. Finished requests are requests that have completed normally or been aborted.

Use this function as follows. If the list of requests to be process 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 in the past, where the list is to start. The maximum is 3 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 when all of the requests have been returned.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |

| Value | Description |
|---|---|
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

### *2.9.16 DIVA_getFilesAndFolders (since version 7.0)*

#### 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
);
```

| Variable | Description |
|---|---|
| **objectName** | Name of the Object to be queried. |
| **objectCategory** | 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** below. |
| **startIndex** | The position in the list to start this iteration. Set at 1 (*one*) to start at the beginning. Values less than 1 are not valid.<br>Set **startIndex** equal to **nextStartIndex** as returned in **DIVA_FILES_AND_FOLDERS** for all subsequent calls. |

| Variable | Description |
|---|---|
| **batchSize** | The maximum size of the returned list of objects. Must be set to a value no greater than 1000; the recommended setting is 500.<br>Note: 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** | Pointer to the returned data. See description of **DIVA_FILES_AND_FOLDERS** below. 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;
```

| List Type | Function will Return |
|---|---|
| **DIVA_LIST_TYPE_FILES_ONLY** | Files only. |
| **DIVA_LIST_TYPE_FOLDERS_ONLY** | Folders: **Valid only for Complex Objects.** |
| **DIVA_LIST_TYPE_FILES_AND_FOLDERS** | Files and Folders: **Valid only for Complex Objects.** |

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

| Variable | Description |
|---|---|
| **objectSummary** | **ID of the DIVArchive Object. See the description below.** |

| Variable | Description |
|---|---|
| **isComplex** | **True** when this 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** | Pointer to the list of files and folders. See description of **DIVA_FILE_FOLDER_INFO** below. |

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

| Variable | Description |
|---|---|
| **objectName** | The Object Name. |
| **objectCategory** | The Object Category. |

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

| Variable | Description |
|---|---|
| **fileOrFolderName** | The File or Folder Name. |
| **isDirectory** | **True** if the component is a directory and not a file name. |
| **sizeBytes** | The size of the file in bytes. Valid only for files. |

| Variable | Description |
|---|---|
| **fileId** | This is a unique ID for each file that is created by DIVArchive as part of the processing of this command. |
| **totalNumFilesFolders** | The number of files and sub folders. **Valid only for folders in a Complex Object.** |
| **totalSizeFilesFolders** | The total size of all files (*including sub folders*). **Valid only for folders in a Complex Object.** |
| **pChecksumInfoList** | 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 description below**.** |

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

| Variable | Description |
|---|---|
| **checksumType** | The type of the checksum, MD5, SHA1 etc. |
| **checksumValue** | The value of the checksum, in hexadecimal string format. |
| **isGenuine** | **True** if this checksum was provided at the time of the archive and verified as Genuine. |

## Description

Get 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.

In order to get all of the file and folder names for an object, the user sets the **startIndex** to zero (0). A list of names of size specified is returned. The user then sets **startIndex** to the value of **nextStartIndex** and again makes the function call. Continue this until the return value = **DIVA_WARN_NO_MORE_OBJECTS**.

## Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |
| **DIVA_WARN_NO_MORE_OBJECTS** | The end of the list has been reached during the call. |

### 2.9.17 DIVA_getGroupsList

#### Synopsis

```
#include "DIVAapi.h"

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

| Variable | Description |
|---|---|
| **groups** | Pointer to a list of **DIVA_GROUP_DESC** structures. |

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

| Variable | Description |
|---|---|
| **group_name** | The configured name of the tape group. |
| **group_desc** | The group description. |
| **mediaFormatId** | The data format of the tapes added to this group (**DIVA_MEDIA_FORMAT_LEGACY, DIVA_MEDIA_FORMAT_AXF,** *or* ***DIVA_MEDIA_FORMAT_AXF_10***). Refer to Table 1: Definitions, Acronyms, and Special Terms for more format information. |

#### Description

Returns the description of all of the groups.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

**See Also:** **DIVA_getObjectInfo**

### 2.9.18 DIVA_getObjectDetailsList

#### 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
);
```

| Variable | Description |
|---|---|
| **fFirstTime** | The first time this function is called, this parameter must be set to true.<br>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**. |
| **intialTime** | The start time of the list. Data will be collected and returned corresponding to this time and later. To retrieve all items in the database, use a time of zero (0). |
| **pListType** | One of the codes defined by the enumeration **DIVA_LIST_TYPE** shown below. |

| Variable | Description |
|---|---|
| **pObjectsListType** | One of the codes defined by the enumeration **DIVA_OBJECTS_LIST_TYPE** shown below.<br><br>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.<br><br>To retrieve tape-related information for all objects that have created, deleted, repacked, ejected, and/or inserted tape instances since a certain time, set this parameter to **DIVA_INSTANCE_CREATED**, **DIVA_INSTANCE_DELETED**, **DIVA_INSTANCE_REPACKED**, **DIVA_INSTANCE_EJECTED**, **DIVA_INSTANCE_INSERTED**, respectively.<br><br>To retrieve any combination of the above, use the '**\|**' (*pipe*) operator.<br><br>**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. Must be set to a value no greater than 1000; the recommended setting is 500.<br><br>Note: 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 ***** wildcard can be used (*e.g.: "*video"*). |
| **pMediaName** | Filter the returned list of objects based on the provided Media Name. The ***** wildcard can be used (*e.g.: "soap*"*). |
| **pLevelOfDetail** | One of the codes defined by the enumeration **DIVA_LEVEL_OF_DETAIL** (*see below*). Filtering by Object Name, Category, and Group (*Media Name*) is performed at all levels of detail.<br><br>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.<br><br>The **DIVA_TAPE_INFO_LIST** only works with the **DIVA_OBJECTNAME_AND_CATEGORY** and **DIVA_INSTANCE** level of detail. |

| Variable | Description |
|---|---|
| **listPosition** | A vector of **DIVA_STRING** type. The elements of this list are for internal use only and need not be extracted by the user.<br><br>When **pFirstTime** is true, a new empty list must be constructed and included.<br><br>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** | Pointer to **DIVA_OBJECT_DETAILS_LIST** class (*see below*). This is the output parameter that will contain the response to the call.<br><br>Use the **listPosition** parameter from this response as the **listPosition** argument in subsequent calls to **GetObjectDetailsList**.<br><br>For **pListType = DIVA_OBJECTS_LIST**, all of the object and/or instance information is stored in the **objectInfo** attribute.<br><br>For **pListType = DIVA_TAPE_INFO_LIST**, all of the 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;
```

| Level of Detail Setting | Return Value |
|---|---|
| **DIVA_OBJECTNAME_AND_CATEGORY (0)** | **getObjectDetailsList** function will only return the Object Name and Category. |
| **DIVA_MISC (1)** | **getObjectDetailsList** function will return the comments, archive date, and name and path on the source, in addition to all data returned with the **DIVA_OBJECTNAME_AND_CATEGORY** level of detail. |
| **DIVA_COMPONENT (2)** | **getObjectDetailsList** function will return the size of the object, list of components value in addition to all data returned with the **DIVA_MISC** level of details. |

| Level of Detail Setting | Return Value |
|---|---|
| **DIVA_INSTANCE (3)** | **getObjectDetailsList** function will return all instance information, repack state, and related active request information data, in addition to 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;
};
```

| Variable | Description |
|---|---|
| **listType** | One of the codes defined by the enumeration **DIVA_LIST_TYPE**. |
| **siteId** | The DIVArchive System Name as configured in **manager.conf** |

| Variable | Description |
|---|---|
| **listPosition** | After the first (*and any subsequent*) call, 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** | Pointer to a **DIVA_OBJECT_INFO** structure (*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** | Pointer to a list of **DIVA_OBJECT_TAPE_INFO** structures (*the structure should be allocated and deleted by the caller*). 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;
int                                  objectSize;
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;
};
```

| Variable | Description |
|---|---|
| **objectSummary** | The Object Name and Category. |
| **UUID** | Universally Unique Identifier to uniquely identify each object created in DIVArchive across all Oracle customer sites, except for objects created via “Copy As” requests. An object created via a “Copy As” Request will contain the same UUID as that of the source object. |
| **lockStatus** | The locking status of the object. Objects in the archive may be locked. When locked, these objects may not be restored or copied to a new name. This feature is used to prevent the use of an object that has an expired copyright, etc. When this value is 0, the object is unlocked. |
| **objectSize** | The size in Kilobytes. |
| **filesList** | List of the files of the Object. A single wrapper filename is returned for Complex Objects. |
| **objectComments** | Comments saved when the object was archived. |
| **archiveDate** | Seconds since 1970/01/01. |
| **isInserted** | 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** | A list of Object Instances saved to tape. |
| **actorInstances** | A list of Object Instances saved to disk. |
| **objectSource** | The source server used to archive the Object. |
| **rootDirectory** | Root of the object files on **objectsource.** |
| **relatedRequests** | Non-terminated requests. |
| **toBeRepacked** | False unless all instances are going to be repacked. |
| **modifiedOrDeleted** | One of **DIVA_MODIFIED_OR_DELETED**<br>**UNDEFINED** - The **levelOfDetail** does not equal **DIVA_INSTANCE**.<br>**DIVA_CREATED_OR_MODIFIED** – The object was created or an instance was added or removed.<br>**DIVA_DELETED** – The object has been removed. |

| Variable | Description |
|---|---|
| **isComplex** | **True** if this is a Complex Object. |
| **nbFilesInComplexComponent** | Number of files in the Object. **Use only for Complex Objects.** The value is 0 for non-complex objects. |
| **nbFoldersInComplexComponent** | Number of folders in the Object. **Use only for Complex Objects.** The value is 0 for non-complex objects. |

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

| Variable | Description |
|---|---|
| **objectName** | The Object Name. |
| **objectCategory** | The Object Category. |

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

| Variable | Description |
|---|---|
| **instanceId** | Numeric ID. |
| **groupName** | The name of the group to which this tape is assigned. |
| **tapeDesc** | Additional information about this tape. |
| **isInserted** | 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. |

| Variable | Description |
|---|---|
| **reqStatus** | Is the instance Required/Released?<br>**DIVA_REQUIRED** - Instance is requested to be inserted into the library<br>**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;
};
```

| Variable | Description |
|---|---|
| **vsn** | Volume serial number (*barcode*). |
| **isInserted** | 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** | False unless all instances are going to be repacked. |
| **mediaFormatId** | The data format to be used (**DIVA_MEDIA_FORMAT_DEFAULT**, **DIVA_MEDIA_FORMAT_LEGACY, DIVA_MEDIA_FORMAT_AXF,** *or* ***DIVA_MEDIA_FORMAT_AXF_10***). This is used only the when the **List Type** is **Tape**. Refer to Table 1: Definitions, Acronyms, and Special Terms for more format information. |

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

| Variable | Description |
|---|---|
| **instanceID** | Numeric ID. |
| **Actor** | This field reports the Name of the Array of Disks where the instance is stored instead of the Name of an Actor. |

```
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;
```

## Description

The **DIVA_getObjectDetailsList** is an API call used 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 “some time”.
- The *deleted-since* call retrieves all objects deleted-since “some time”.
- If starting from a user-specified time of 0, the *modified-since* call retrieves all objects created since “some time”, and it is simply returning the state of the database from time 0.
- If starting from a user-specified time greater than 0, the call returns all objects created and deleted since “some time”, in addition to all objects with newly created and/or deleted instances.

**Note: The `listPosition` vector returned by a `GetObjectDetailsList` call must be passed into a subsequent call. Its content must not be altered by the user of the call.**

Different levels of details may be specified (*please see Level of Detail Setting Table above*). Level 0 would be the fastest, while Level 3 would return all possible details.

**Note: Only the highest level of detail is supported. Using a lower level of detail will still return all information for objects.**

The output may be structured by the Objects Lists (***DIVA_OBJECTS_LIST** option*) or by Storage Tape (***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 the list of objects (*Object Instances*) added to DIVArchive.
- To retrieve the list of objects (*Object Instances*) deleted from DIVArchive.
- To retrieve the combined list of all changes in DIVArchive Object Database (*adding and deleting objects, adding and deleting instances*).
- To continuously monitor the DIVArchive System to retrieve events of adding/deleting objects and adding/deleting instances.

The API Client application should use **DIVA_TAPE_INFO_LIST** setting in the following cases:

- To retrieve a list of Tape Instance information for added instances.
- To retrieve a list of Tape Instance information for deleted instances.
- To retrieve a list of Tape Instance information for repacked instances.
- To retrieve a list of Tape Instance information for ejected instances.
- To retrieve a list of Tape Instance information for inserted instances**.**

**Note: If all objects are deleted, DIVA_TAPE_INFO_LIST will not return any results for deleted instances.**

### Return Values

Depending upon the input parameters, the **DIVA_getObjectDetailsList** function will return values as described in the table below:

| 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_DELETED_SINCE** | Only **DIVA_OBJECTNAME_AND_CATEGORY** | List Objects that have been deleted since a specified time. |
| | **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.<br>Note: If the list of instances is EMPTY, objects were DELETED. |

| List Type | Objects List Type | Supported Detail Level | Return Value |
|---|---|---|---|
| | | | If the list of instances is NOT EMPTY, objects were CREATED or UPDATED. |
| **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_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_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_INSTANCE_REPACKED (0x0040)** | Only **DIVA_OBJECTNAME_AND_CATEGORY** and **DIVA_INSTANCE** level. | List Objects and Tape information for all Tape Instances repacked since a specified time. |
| | **DIVA_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_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. |

#### 2.9.18.1 Usage with DIVAnet Access Gateway

All filters are applied at an object level:

- If a user requests objects satisfying certain filter constraints, those constraints are applied to the object and not to individual instances of an object.
- If a user specifies 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 not defined at an object level, but rather at an instance 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 a user requests 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 of Object A was added at time 101.
  - The instance that was added has a media name of CAR.
  - Altogether Object A has two instances:
    1. One instance has a media name of TRUCK; the other has a media name of CAR.
- An instance of Object B was removed at time 101.
  - The instance that was removed had a media name of CAR.
  - Altogether Object B has one instance
- A new instance of Object C was added at time 99.
  - The instance that was added has a media name of TRAIN.
  - Altogether Object C has two instances:
    2. One instance has a media name of TRAIN; the other has a media name of HANG GLIDE.
- A user executes a **`getObjectDetailsList`** call with **`MODIFIED SINCE TIME 100`** with **`MEDIA NAME FILTER = T*`**.
  - The only object that was modified since time 100 and has at least once instance with a media name that starts with T is Object A.
- The list returned by the **`getObjectDetailsList`** call in this case is *Object A*.

#### 2.9.18.2 Usage and Recommended Practices

**Sequence of Actions:** It is recommended 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 (0) to include all objects in the database.
4. Create a variable of Boolean type and set to *true* in order to indicate that this is the first call in a sequence of calls.
5. Create variables of Integer type to hold the List Type and Objects List Type 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 mentioned above.
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. Repeat steps 8, 9, & 10 for as long as you need to monitor the DIVArchive System.
13. All variables should be de-allocated after exiting the loop.

**Multiple simultaneous calls** to **DIVA_getObjectDetailsList** are supported. However, since this call places a heavy demand on the database, simultaneous and/or frequent calls to this function should be avoided.

**Continuous monitoring of DIVArchive** requires a procedure similar to the one defined below in the *Continuous Updates Notification Design Pattern* section.

**Duplication of Objects may occur** across different return portions, so it is important to handle this case by examining the data returned by the call. In the case of a **MODIFIED_SINCE** call, it is important to compare the instances of the duplicate object returned by successive calls to see if 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 frequency of the calls should be reduced once you receive an empty list to reduce the load on the DIVArchive Database.

The **`DIVA_getObjectDetailsList`** function can effectively be used for the **Initial Database Synchronization** (*in case the client application is maintaining a database*) and for **Continuous Monitoring.** The same application can use the call for the purpose of the *Initial Database Synchronization*, and later it can be used for the purpose of *Continuous Monitoring* when the database is updated.

During the **Initial Database Synchronization** phase, the application should make frequent sequential calls to synchronize the local database with the DIVArchive Database. The application should call **`DIVA_getObjectDetailsList`**, wait for a response, and then repeat the process.

After the Synchronization Phase, the application should go into the **Continuous Monitoring Phase**, where it should make periodic calls to update the system with the latest object information. If it is possible for the client application, it is recommended that a call interval of *once per several minutes* be used. Continuous frequent execution of this call may impact the database heavily and degrade system performance.

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

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

The *Design Pattern for Continuous Updates Notification* is used in multiple applications and is important in using the DIVArchive API. The client application may use the internal database with the goal of continuously updating the local database information with the changes in the DIVArchive Database. Following the design pattern can help to develop the performance-optimized Updates Notification Workflow.

The application should submit the call with the **Object List Type** set to **`MODIFIED_SINCE`** and set the level of detail required to collect instance-level information. In addition, the first time flag must be set to true and all necessary filter parameters must be set (*Object Name and Category*).

1. The application will receive a list of Objects and a new **`listPosition`**.
2. The next time, the application should execute the call with the **`listPosition`** obtained from Step 1 and the first time flag set to false. If the system is being used solely for synchronization purposes, then it is acceptable to submit another call immediately after receiving the list. 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, then the list will be *EMPTY*, which means that there were no updates since the last call. The application should wait for a specific amount of time, and then retry again.

The application should check the list of instances:

   a. If the value of **modifiedOrDeleted** in the **DIVA_OBJECT_INFO** equals *DELETED*, objects were *DELETED* and the database should be updated.

   b. If the value of **modifiedOrDeleted** in the **DIVA_OBJECT_INFO** equals **CREATED_OR_MODIFIED**, the object was *CREATED* or *UPDATED*.

      i. If the Object existed in the database previously, the database should be updated with the list of instances.

      ii. If the Object does not exist in the database, it should be added to the database.

**Note: To ensure continuous updates, the `listPosition` object should be preserved for 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
```

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

| Value | Description |
|---|---|
| **DIVA_WARN_NO_MORE_OBJECTS** | The end of the list has been reached during the call (*see Description*). |

### *2.9.19 DIVA_getObjectInfo*

#### Synopsis

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

**Note: The `vector<DIVA_ACTOR_INSTANCE_DESC> *actorInstances` parameter is kept unchanged for compatibility, although it is formally a vector of Disk Instances and not Actor Instances.**

**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.**

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

| Variable | Description |
|---|---|
| **objectName** | Name of the Object to be queried. |
| **objectCategory** | Category assigned to the Object when it was archived. This parameter can be a null string (*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 description of **DIVA_OBJECT_INFO**. |

#### Description

Returns information about a particular Object in the DIVArchive System.

#### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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**
- **DIVA_deleteObject**

### *2.9.20 DIVA_getPartialRestoreRequestInfo*

#### Synopsis

```
#include "DIVAapi.h"

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

| Variable | Description |
|---|---|
| **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 Filename, a vector of the offsets used for the transfer, and a Destination Filename.<br>This vector should 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. |

#### Description

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

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

1. The request number to be queried must be a Partial File Restore Request that has been **successfully** completed.

2. 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.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |
| **DIVA_ERR_NO_SUCH_REQUEST** | **requestNumber** identifies no request. |
| **DIVA_ERR_INVALID_PARAMETER** | **requestNumber** identifies no completed Partial File Restore Request. |

**See Also:**

- **DIVA_partialRestoreObject**
- **DIVA_getRequestInfo**

### 2.9.21 DIVA_getRequestInfo

#### Synopsis

```
#include “DIVAapi.h”

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

| Variable | Description |
|---|---|
| **requestNumber** | Identifies the request to be queried. |
| **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
};
```

| Parameter | Description |
|---|---|
| **requestNumber** | DIVArchive Request Number. |
| **requestType** | See definition of **DIVA_REQUEST_TYPE** below |
| **requestState** | See definition of **DIVA_REQUEST_STATE** below |
| **progress** | From 0 to 100 (%)if **requestState** is **DIVA_TRANSFERRING** or **DIVA_MIGRATING**. |
| **abortionReason** | If **requestState** is **DIVA_ABORTED**, else 0**.** |
| **objectSummary** | See definition of **DIVA_OBJECT_SUMMARY** below |
| **repackTapes** | If the **requestType** is repack**.** |
| **additionalInfo** | See 2.9.21.1 for more information on the use of this field. |
| **submissionDate** | The date and time the request was submitted. This is UTC time in seconds (*seconds since 1970/01/01*). |
| **completionDate** | The date and time the request completed. This is UTC time in seconds. This value will be -1 if the request has not yet completed. |

```
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;
```

| Parameter | Description |
|---|---|
| **DIVA_AR_NONE = 0** | Request not aborted. |
| **DIVA_AR_DRIVE** | Drive trouble. |
| **DIVA_AR_TAPE** | Tape trouble. |
| **DIVA_AR_ACTOR** | Actor trouble. |
| **DIVA_AR_DISK** | Disk trouble. |
| **DIVA_AR_DISK_FULL** | Disk is full. |
| **DIVA_AR_SOURCE_DEST** | Source/Destination trouble. |
| **DIVA_AR_RESOURCES** | Resources attribution troubles |
| **DIVA_AR_LIBRARY** | Library trouble. |
| **DIVA_AR_PARAMETERS** | Wrong request parameters. |
| **DIVA_AR_UNKNOWN** | Unknown code. |
| **DIVA_AR_INTERNAL** | Internal DIVArchive Manager error. |
| **DIVA_AR_SOURCE_DEST2** | Deprecated (*left for software compatibility*). |

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

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

| Variable | Description |
|---|---|
| **objectName** | The Object Name. |
| **objectCategory** | The Object Category. |

### Description

Obtains information about an Archive, Restore, Delete, or Repack Request.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |
| **DIVA_ERR_NO_SUCH_REQUEST** | **requestNumber** identifies no request. |

#### *2.9.21.1 Additional_Info*

The **Additional_Info** field of the **DIVA_REQUEST_INFO** structure may contain one or more of the following, depending upon 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.

### MOB ID Sample:

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

### XML Document:

Depending upon 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.

```
<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 Clip ID is included when the request was for a Restore to a Quantel device:

An ISA gateway never overwrites clips. A new Clip ID is created for every imported clip. The **ClipID** of the created clip will be supplied at the end of the Transfer Complete message.

```
226 Transfer Complete. [new ClipID]
```

The Actor captures this new **ClipID** at the end of the transfer and forwards it to the Manager. In order 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 shown here:

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

### 2.9.22 DIVA_getSourceDestinationList

#### Synopsis

```
#include “DIVAapi.h”

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

| Variable | Description |
|---|---|
| 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
} 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;

};
```

| Parameter | Description |
|---|---|
| server_Address | Server IP Address. |
| server_ConnectOption | Server connection options. |
| server_MaxAccess | Maximum number of accesses to the server. |
| server_MaxReadAccess | Maximum number of read accesses to the server. |
| server_MaxThroughput | Server Maximum Throughput. |
| server_MaxWriteAccess | Server maximum write access. |
| server_Name | Server Name. |
| Server_ProductionSystem | Server Production System Name. |
| server_RootPath | Server Root Path. |
| server_SourceType | Server Source Type. |

### Description

The purpose of this function is to provide a list of Source Servers present in a particular DIVArchive System.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **DIVA_ERR_BROKEN_CONNECTION** | The connection with the DIVArchive Manager has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

### 2.9.23 DIVA_getStoragePlanList

#### Synopsis

```
#include "DIVAapi.h"

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

| Variable | Description |
|---|---|
| **spList** | Pointer to a list of Storage Plan Names. |

#### Description

This function returns the list of Storage Plan Names that are defined in the DIVArchive System.

#### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |

| Value | Description |
|---|---|
| DIVA_ERR_INTERNAL | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

### 2.9.24 DIVA_getTapeInfo

#### Synopsis

```
#include “DIVAapi.h”

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

| Variable | Description |
|---|---|
| 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;
long       remainingSize ;
long       totalSize ;
bool       isInserted;
string     externalizationComment;
bool       isGoingToBeRepacked;
int        mediaFormatId;
};
```

| Parameter | Description |
|---|---|
| **setID** | **Equivalent to pool number in release 4.2.** |
| **typeID** | Tape Type ID |
| **type** | Tape Type Name |
| **fillingRatio** | **(last written block)/(total count of blocks)** |
| **fragmentationRatio** | **1-(count of valid blocks)/(last written block).**<br>Valid blocks are blocks used for archived objects which are not currently deleted. |
| **mediaFormatId** | **DIVA_MEDIA_FORMAT_DEFAULT, DIVA_MEDIA_FORMAT_LEGACY, DIVA_MEDIA_FORMAT_AXF,** *or* ***DIVA_MEDIA_FORMAT_AXF_10.***<br>Refer to Table 1: Definitions, Acronyms, and Special Terms for more format information. |

### Description

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

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |

| Value | Description |
|---|---|
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |
| **DIVA_ERR_TAPE_DOESNT_EXIST** | There is no tape associated with the given barcode. |

### *2.9.25 DIVA_insertTape*

#### 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
);
```

| Variable | Description |
|---|---|
| **Require** | When true, perform a **DIVA_require()** on every instance located on the successfully inserted tapes. |

| Variable | Description |
|---|---|
| **PriorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY**. With this value the Manager uses the default priority for this request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] or predefined values*) yields a **DIVA_ERR_INVALID_PARAMETER** error. |
| **acsId**<br>(*in the second form only*) | Numeric ID of the ACS where the Insert operation should be executed.<br>When **acsId = -1** (*default used for the first form*), the Insert attempt will be performed in all known ACSs. |
| **capId**<br>(*in the second form only*) | Numeric ID of the CAP from which tapes will be inserted.<br>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** | Number identifying the request. |

### Description

Submits an *Enter Request* to DIVArchive. This request completes when the operator has entered some tapes into the library. The application is responsible for managing which tapes need to be entered.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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** | 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**.

### 2.9.26 DIVA_linkObjects

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_linkObjects (
DIVA_STRING          parentName,
DIVA_STRING          parentCategory,
DIVA_STRING          childName,
DIVA_STRING          childCategory,
bool                 cascadeDelete,
bool                 cascadeRestore
);
```

| Variable | Description |
|---|---|
| **parentName** | Parent Object Name. |
| **parentCategory** | Parent Object Category. |
| **childName** | Child Object Name. |
| **childCategory** | 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. |

#### Description

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.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been 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 has not been understood by the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

## *2.9.27 DIVA_lockObject*

### Synopsis

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

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

```
DIVA_STATUS DIVA_lockObject (
IN DIVA_STRING   objectNme,
IN DIVA_STRING   category,
IN string        options
);
```

| Variable | Description |
|---|---|
| **objectName** | Name of the object. |
| **category** | The Category to which the object was assigned when archived. |
| **options** | TBD |

### Return Values

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

## *2.9.28 DIVA_multipleRestoreObject*

### 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;
```

| Value | Description |
|---|---|
| **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 (*this may result in an error if several objects have the same name*). |
| **destinations** | List of Destinations (*e.g. video server or browsing server*) available to put the object files. The names must be known by the DIVArchive Configuration Description.<br><br>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:<br><br>**DIVA_QOS_DEFAULT:** restoring is performed according to the default Quality Of Service (*currently: direct and cache for restore operations*).<br><br>**DIVA_QOS_CACHE_ONLY:** Use cache restore only.<br><br>**DIVA_QOS_DIRECT_ONLY:** Use direct restore only.<br><br>**DIVA_QOS_CACHE_AND_DIRECT:** Use cache restore if available or direct restore if cache restore is not available.<br><br>**DIVA_QOS_DIRECT_AND_CACHE:** Use direct restore if available or cache restore if direct restore is not available.<br><br>**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.<br><br>**DIVA_QOS_NEARLINE_AND_DIRECT:** Use Nearline restore if available, or direct restore if Nearline restore is not available.<br><br>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:<br><br>**DIVA_RESTORE_SERVICE_DO_NOT_OVERWRITE:** Do not overwrite existing files on the destination server. |

| Value | Description |
|---|---|
| **priorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY**. With this value the Manager uses the default priority for this request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] 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:<br>• A null string to specify no options.<br>• **-login**: Login used for some sources. This option obsoletes the **-gateway** option of the previous version.<br>• **-pass**: Password used in conjunction with the **-login** option for some sources. |
| **requestNumber** | Request Number assigned to this request. This number is used for querying the status or cancelling this request. |

## Description

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. To check that the operation was successful the application must call the function **DIVA_getRequestInfo()**.

**Note:**

- **If DIVA_MultipleRestoreObject() is launched with a single destination, the restore is automatically converted to a DIVA_RestoreObject().**
- **The Request will continue even if an error occurs with one of the destinations.**

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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** | 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 (*being Archived, Restored, Deleted, etc.*). |

| Value | Description |
| --- | --- |
| **DIVA_ERR_SOURCE** or **DESTINATION_DOESNT_EXIST** | The specified source is not known by the DIVArchive System. |
| **DIVA_ERR_OBJECT_PARTIALLY_DELETED** | The specified object has instances that are partially deleted. |

**See Also:**

- **DIVA_restoreObject**
- **DIVA_getRequestInfo**
- **DIVA_copyToGroup**

### 2.9.29 DIVA_partialRestoreObject

#### 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
);
```

| Value | Description |
|---|---|
| **objectName** | 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 (*this may result in an error if several objects have the same name*). |
| **instanceID** | The ID of a non-spanned Tape Instance, or **DIVA_ANY_INSTANCE**. |
| **Filelist** | List of the files of the object to be Partially Restored. Each structure contains the Source File Name, a vector of offset pairs, and a Destination File Name. The same source file may be used in several structures, but destination files must be unique. A file present in the DIVArchive Object may not be in any structure (*otherwise it won't be restored*). |
| **destination** | Destination (*e.g. video server or browsing server*) to put the object files. This name must be known by the DIVArchive Configuration Description. |
| **filesPathRoot** | Root folder on the destination where the object files will be placed. If null (***string("")***), the files will be placed in the **FILES_PATH_ROOT** folder specified when archiving the object (*using the **DIVA_archiveObject()** function*). |

| Value | Description |
|---|---|
| **qualityOfService** | One of the following codes:<br>**DIVA_QOS_DEFAULT:** restoring is performed according to the default Quality Of Service (*currently: direct restore)*.<br>**DIVA_QOS_CACHE_ONLY:** Use cache restore only.<br>**DIVA_QOS_DIRECT_ONLY:** Use direct restore only.<br>**DIVA_QOS_CACHE_AND_DIRECT:** Use cache restore if available or direct restore if cache restore is not available.<br>**DIVA_QOS_DIRECT_AND_CACHE:** Use direct restore if available or cache restore if direct restore is not available.<br>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:<br>**DIVA_RESTORE_SERVICE_DO_NOT_OVERWRITE:** Do not overwrite existing files on the destination server. |
| **priorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY**. With this value the Manager uses the default priority for this request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] 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:<br>• A null string to specify no options.<br>• **-login:** Login is used for some sources. This option obsoletes the **-gateway** option of the previous version.<br>• **-pass:** Password used in conjunction with the **-login** option for some sources. |

| Value | Description |
|---|---|
| **format** | **DIVA_FORMAT_BYTES**<br>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, the destination file is created by concatenating every corresponding extract.<br>**DIVA_FORMAT_BYTES_HEADER**<br>Deprecated, left for compatibility purposes only.<br>**DIVA_FORMAT_VIDEO_GXF**<br>Offsets must be given as time codes.<br>The file to be partially restored is expected to be in **GXF** format.<br>The **fileList** vector parameter is expected to contain only one **DIVA_OFFSET_SOURCE_DEST** element as well as the **offsetVector** vector, which is expected to contain only one **DIVA_OFFSET_PAIR** element.<br>Only the **DIVA_QOS_DIRECT_ONLY** Quality Of Service is supported for this format.<br>**DIVA_FORMAT_VIDEO_SEA**<br>Offsets must be given as time codes.<br>The file to be partially restored is expected to be in **SAF** format and provide an index file.<br>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*).<br>• The source filename in each structure must have the **.wav** or the **.WAV** extension. |

| Value | Description |
|---|---|
| **format (*continued*)** | • Each structure must contain exactly one **DIVA_OFFSET_PAIR** structure with a time code pair equal to the time code pair associated with the **AVI** file.<br>• The next part is delimited by the first **DIVA_OFFSET_SOURCE_DEST** structure associated with an **AVI** file.<br>• 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*).<br>**DIVA_FORMAT_VIDEO_MPEG2_TS**<br>Offsets must be given as time codes.<br>Video file must be encoded using the **MPEG2** Transport Stream format. Use this for VELA encoders.<br>**DIVA_FORMAT_VIDEO_MXF**<br>Offsets must be given as time codes.<br>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.<br>**DIVA_FORMAT_VIDEO_PINNACLE**<br>Offsets must be given as time codes.<br>This Partial File Restore format expects a specific object structure. It is applicable to Pinnacle clips composed of three files, **header**, **ft,** and **std**. The MSS Source/Destination Type is preferred to create this clip with DIVArchive.<br>The **fileList** vector parameter is expected to contain only one **DIVA_OFFSET_SOURCE_DEST** element, as well as the **offsetVector** vector, which is expected to 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. |

| Value | Description |
|---|---|
| **format (*continued*)** | **DIVA_FORMAT_VIDEO_OMNEON**<br>Offsets must be given as time codes.<br>This type of Partial File Restore can be used to partially restore Quicktime files (*referenced and self-contained clips are supported*). A detailed matrix of supported Quicktime clips is given in the product description.<br>The **fileList** vector parameter is expected to contain only one **DIVA_OFFSET_SOURCE_DEST** element as well as the **offsetVector** vector, which is expected to 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.<br>**DIVA_FORMAT_VIDEO_LEITCH**<br>Offsets must be given as time codes.<br>Video file must be encoded using the LEITCH Video Server and the format is **LXF**.<br>**DIVA_FORMAT_VIDEO_QUANTEL**<br>Offsets must be given as time codes.<br>This type of Partial File Restore can be used to partially restore Quantel clips that have been archived with a **QUANTEL_QCP** Source/Destination Type.<br>**DIVA_FORMAT_AUTODETECT**<br>Offsets must be given as time codes.<br>This type of Partial File Restore can detect video clips with the following archive formats:<br>• QuickTime self-contained.<br>• QuickTime with referenced media files (*The **.mov** file must be in the first position*).<br>• **DIF + WAV** files.<br>• **AVI** with audio interleaved (*separated **WAV** are not supported at this time*).<br>• **MXF** (*self-contained*)<br>• **MPEG PS**<br>• **LXF**<br>• **Seachange** (*The **.pd** file must be in the first position*) |

| Value | Description |
|---|---|
| **format (*continued*)** | The **fileList** vector parameter is expected to contain only one **DIVA_OFFSET_SOURCE_DEST** element as well as the **offsetVector** vector, which is expected to contain only one **DIVA_OFFSET_PAIR** element. The **DIVA_OFFSET_SOURCE_DEST** element must be associated with:<br>• The **.mov** file if it is a Quicktime clip.<br>• The **.dif** file if it is a **dv** file.<br>• The **.avi** file if it is an **AVI** clip.<br>**DIVA_FORMAT_FOLDER_BASED**<br>Specifies a set of files and folders to be restored. A recursive flag may be set to restore subfolders. All specified files and folders are restored.<br>**DIVA_FORMAT_DPX**<br>Specifies a set of intervals, frame X through frame Y, where frames are sorted and tranversed alphanumericly.<br>Only files with **.tif** or **.tiff** data formats are supported. All files must have an extension of **.dpx**. The first frame of a DPX Object is Frame 1. Frame numbers of 0 and -1 may be used to refer to the first and last frame. |
| **requestNumber** | Request Number assigned to this request. This number is used for querying the status or cancelling 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;
};
```

| Value | Description |
|---|---|
| **sourceFile** | The source filename when format is other than **DIVA_FORMAT_FOLDER_BASED** or **DIVA_FORMAT_DPX.** |
| **offsetVector** | 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 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.
```

| Constructor | Description |
|---|---|
| **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 time code offsets. Timecodes are formatted as HH:MM:SS:FF. |

| Attribute Accessors | Description |
|---|---|
| **DIVA_SPEC bool isTimeCode();** | True if the offset pair was constructed with time code offsets. |
| **DIVA_SPEC DIVA_STRING getTimeCodeBegin();** | Return the begin offset as a time code. |
| **DIVA_SPEC DIVA_STRING getTimeCodeEnd();** | Return the end offset as a time code. |
| **DIVA_SPEC __int64 getByteBegin();** | Return the begin offset as bytes. |

| Attribute Accessors | Description |
|---|---|
| `DIVA_SPEC __int64 getByteEnd();` | Return the end offset as bytes. |

```
class DIVA_FILE_FOLDER {
public:
      DIVA_STRING          fileFolder;
      DIVA_STRING          option
};
```

| Variable | Description |
|---|---|
| `FileFolder` | The file or folder name. |
| `Option` | Option (*Ex: `-r` to recurse folders*). |

```
class DIVA_RANGE {
public:
      int                  startRange;
      int                  endRange;
    };
```

| Variable | Description |
|---|---|
| `startRange` | The first frame number to be restored. |
| `endRange` | The last frame number. |

// The format gives information about how to interpret the interval and about which specific operation should performed eventually.

```
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 ;
```

| Value | Description |
|---|---|
| **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_MEPEG_TS** | MPEG Transport Stream |
| **DIVA_FORMAT_VIDEO_MXF** | MXF video format |
| **DIVA_FORMAT_VIDEO_PINNICLE** | 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 detection of the format. |
| **DIVA_FORMAT_FOLDER_BASED** | Fully restore the specified files and/or folders. |
| **DIVA_FORMAT_DPX** | DPX video format. |

## Description

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 function **`DIVA_getRequestInfo()`**.

If the request was not accepted (*e.g. if the requested object is on media that is not available*) the request will generate an error. 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.

The Manager will use the **instanceID** field to select the instance of the object to be used for the Partial Restore operation. If **DIVA_ANY_INSTANCE** is used, the Manager will choose an appropriate instance to be restored.

DIVArchive supports four types of Partial Restore: 1) Byte Offset, 2) Timecode, 3) Files and Folders, 4) DPX. The type of Partial Restore that will be implemented is determined by the format parameter in the request.

The following describes each type of Partial Restore:

- **Byte Offset (*format equals DIVA_FORMAT_BYTES*):** This allows a range of bytes to be extracted from a particular file in the archive. For instance, it is possible to 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**.

  **Note: The result of the Byte Offset Partial Restore is usually unplayable when applying to video files. Actor will not apply the header, footer, etc. 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 this object, specify the **sourceFile** in the archive and what you would like to call 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 (*format equals DIVA_FORMAT_VIDEO_*)*:** This type of Partial Restore allows you to select a portion of a particular media file based on timecode. For instance, 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**. This file is a smaller version of the original movie file.

  **Note: The result of the Timecode Partial Restore is a valid clip when applying to video files. Actor will apply the header, footer, etc. according to the video format. If Actor is unable to parse the format, the request will be aborted. 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 to be partially restored. For example, if the file to be 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. To use auto-detect, specify **DIVA_FORMAT_AUTODETECT** in the **format** field.

  Create a **DIVA_OFFSET_SOURCE_DEST** object (*in the **fileList** parameter of the request*). In this object, add a **DIVA_OFFSET_PAIR** object (*the **offsetVector** parameter*) containing 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 (*format equals* `DIVA_FORMAT_FOLDER_BASED`):** This type of Partial Restore allows extracting entire files from the archive or extracting entire directories and their contents. DIVArchive allows you to extract multiple files and directories in the same request. The files are restored with the file/path names that were specified in the archive; no renaming option is valid in File/Folder 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 File/Folder Partial Restore, all files within that folder (*as well as the folder itself*) are restored as well. In addition, each directory to be restored can have a **`-r`** option to recursively restore all folders nested within the target folder.

  To issue a File/Folder Partial Restore, the **`format`** field in the request should be populated with the value **`DIVA_FORMAT_FOLDER_BASED`**. Create a **`DIVA_OFFSET_SOURCE_DEST`** object (*in the `fileList` parameter of the request*). In this object, add a **`DIVA_FILE_FOLDER`** object (*in the `fileFolder` parameter*) that contains the name of the file or folder to be restored, and any options (*such as the recursive option*) for that directory. **It is important to note that the `offsetVector`, `sourceFile`, `destFile`, and `range` parameters should not be specified!**

- **DPX (*format equal `DIVA_FORMAT_DPX`*):** This Partial Restore type allows extracting a range of DPX files from the archive. In this type of restore, the entire object is viewed as a single media item, with one DPX file representing 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`** file (*or `.tif` or `.tif` file*) in the archived object is considered frame 1, the second **`.dpx`** in the archive is frame 2, and so on.

  For example, if a user wants to extract frames 10 through 15 using DPX Partial Restore, this would restore the 10th **`.dpx`** file that appears in the archive, 11th **`.dpx`** file, etc… ending with the 15th **`.dpx`** file, for a total of six 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:

  - 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 considered invalid.
- Specifying Frame 0 to 0 is currently invalid and will not return the first frame as might be intended.
- Specifying Frame 0 to 1 or Frame 1 to 1 will return the first frame.
- Specifying the Frame -1 in the first frame is currently an error, which does not allow you to specify Frame -1 to -1 to return the exact last frame in the event that the exact number of the last frame is unknown.

**Examples:**

- **`start=0 – end=1`**
  - Restores the first frame only.
- **`start=600 – end=635, start=679 – end=779`**
  - Restores frames 600 through 635, and frames 679 through 779.
- **`start=810 – end=-1`**
  - Restores all frames from frame 810 to the end of the archive.

To issue a DPX Partial Restore, 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, add a **`DIVA_RANGE`** object (i*n the **`range`** parameter*) that contains the start and end frames of the range to be restored. **It is important to note that the `offsetVector`, `sourceFile`, `destFile`, and `fileFolder` parameters should not be specified!** If you wish 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 fashion.

The actual filename may, or may not, match the frame number in DIVArchive. Upon restore, DIVArchive interrogates the archive, finds the file order, and determines the Frame Number from the resulting file order found; it does not consider the filename. The first **`.dpx`**, **`.tif`**, or **`.tiff`** file found is considered Frame 1.

Care must be given when archiving DPX files to ensure they can be partially restored properly. This is 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 upon the order in which the **`.dpx`** files appear in the archive. This order, by default, is based on ordering established by the source and is typically alphanumeric. For example, NTFS DISK Source/Destinations order files and folders case insensitively as a general rule (*but not where diacritical marks, such as* **`'`**, **`` ` ``**, **`^`**, *etc. are applied*).

When DIVArchive encounters a subfolder, by default it recursively processes all of the children of that folder (*including subfolders*) 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 – you may 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, then subdirectories. In DIVArchive 7.3, the Actor allows the archive options **`-file_order DIRS_FIRST`** or **`-file_order FILES_FIRST`** to address these issues.

DPX Partial Restore looks at an 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 via File/Folder Partial Restore, but they will be viewed as one long movie clip to DPX Partial Restore. If this is a desired effect, ensure that the directories are sorted alphanumerically in the order that the frames should be arranged.

The first **`.dpx`** file (*or* ***`.tif`****, or* ***`.tiff`*** *file*) in the archived object is considered frame 1, the second **`.dpx`** in the archive is frame 2, and so on.

DIVArchive does not perform any special audio handling for DPX media (*other than what might be embedded in DPX files themselves*). DIVArchive can support transcoding of DPX media, but keep in mind that a transcoder may change the filenames and/or file order of the DPX archive.

### Return Values

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

| Value | Description |
|---|---|
| **`DIVA_OK`** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **`DIVA_ERR_NOT_CONNECTED`** | No open connection. |
| **`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 has been broken. |
| **`DIVA_ERR_TIMEOUT`** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **`DIVA_API_TIMEOUT`** variable and equals 180 seconds by default. |
| **`DIVA_ERR_UNKNOWN`** | An unknown status has been received from the DIVArchive Manager. |
| **`DIVA_ERR_INTERNAL`** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

| Value | Description |
|---|---|
| **DIVA_ERR_INVALID_PARAMETER** | A parameter value has not been understood by the DIVArchive Manager. |
| **DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS** | Count of simultaneous requests reached the maximum allowed value. This variable is set in the **manager.conf** 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_INSTANCE_OFFLINE** | The instance specified for restoring this object is ejected, or the Actor owning the specified Disk Instance is not available. |
| **DIVA_ERR_INSTANCE_DOESNT_EXIST** | The instance specified for restoring this object does not exist. |
| **DIVA_ERR_OBJECT_IN_USE** | The object is currently in use (*being Archived, Restored, Deleted, etc.*). |
| **DIVA_ERR_SOURCE or DESTINATION_DOESNT_EXIST** | The specified source is not known by the DIVArchive System. |
| **DIVA_ERR_OBJECT_PARTIALLY_DELETED** | The specified object has instances that are Partially Deleted. |

**See Also:**

- **DIVA_restoreObject**
- **DIVA_getRequestInfo**
- **DIVA_getPartialRestoreRequestInfo**

### 2.9.30 DIVA_release

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_release (
IN DIVA_STRING        objectName,
IN DIVA_STRING        categoryName,
IN int                instanceID
);
```

| Variable | Description |
|---|---|
| **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 (*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. |

#### Description

Indicates to the DIVArchive Manager that this instance can be externalized. If this instance has already been released, this function has no effect.

The list of instances that are **RELEASED** and **INSERTED** may be retrieved and shown at the Control GUI.

#### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **DIVA_ERR_SYSTEM_IDLE** | The DIVArchive System is no longer able to accept connections and queries. |

| Value | Description |
|---|---|
| **DIVA_ERR_BROKEN_CONNECTION** | The connection with the DIVArchive Manager has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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_OBJECT_DOESNT_EXIST** | The specified object does not exist in the DIVArchive Database. |
| **DIVA_ERR_INSTANCE_DOESNT_EXIST** | The specified instance does not exist. |
| **DIVA_ERR_INSTANCE_MUST_BE_ON_TAPE** | The specified instance is not a Tape Instance. |
| **DIVA_ERR_NO_INSTANCE_TAPE_EXIST** | No Tape Instance exists for this object. |
| **DIVA_ERR_SEVERAL_OBJECTS** | More than one object with the specified name exists in the DIVArchive Database. |

**See Also:** **DIVA_require**

### 2.9.31 DIVA_require

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_require(
IN DIVA_STRING        objectName,
IN DIVA_STRING        categoryName,
IN int                instanceID
);
```

| Variable | Description |
|---|---|
| **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 (*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. |

#### Description

Indicates to the DIVArchive Manager that this instance should be inserted; If the instance is already inserted, this function has no effect.

The list of instances that are **REQUIRED** and **EJECTED** may be retrieved and shown at the Control GUI.

#### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **DIVA_ERR_SYSTEM_IDLE** | The DIVArchive System is no longer able to accept connections and queries. |

| Value | Description |
|---|---|
| **DIVA_ERR_BROKEN_CONNECTION** | The connection with the DIVArchive Manager has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the DIVA_API_TIMEOUT variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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_OBJECT_DOESNT_EXIST** | The specified object does not exist in the DIVArchive Database. |
| **DIVA_ERR_INSTANCE_DOESNT_EXIST** | The specified instance does not exist. |
| **DIVA_ERR_INSTANCE_MUST_BE_ON_TAPE** | The specified instance is not a Tape Instance. |
| **DIVA_ERR_NO_INSTANCE_TAPE_EXIST** | No Tape Instance exists for this object. |
| **DIVA_ERR_SEVERAL_OBJECTS** | More than one object with the specified name exists in the DIVArchive Database. |

**See Also:** **DIVA_release**

### 2.9.32 DIVA_restoreInstance

#### 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
);
```

| Variable | Description |
|---|---|
| **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 (*this may result in an error if several objects have the same name*). |
| **instanceID** | Instance Identifier. |
| **Destination** | Destination (*e.g. video server or browsing server*) to put the object files. This name must be known by the DIVArchive Configuration Description. |
| **filesPathRoot** | Root folder on the destination in which the object files will be placed. If null (***string("")***), the files will be placed in the **FILES_PATH_ROOT** folder specified when archiving the object (*using the* ***DIVA_archiveObject()*** *function*). |

| Variable | Description |
|---|---|
| **qualityOfService** | One of the following codes:<br>**DIVA_QOS_DEFAULT:** restoring is performed according to the default Quality Of Service (*currently: direct and cache for restore operations*).<br>**DIVA_QOS_CACHE_ONLY:** Use cache restore only.<br>**DIVA_QOS_DIRECT_ONLY:** Use direct restore only.<br>**DIVA_QOS_CACHE_AND_DIRECT:** Use cache restore if available or direct restore if cache restore is not available.<br>**DIVA_QOS_DIRECT_AND_CACHE:** Use direct restore if available or cache restore if direct restore is not available.<br>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:<br>**DIVA_RESTORE_SERVICE_DO_NOT_OVERWRITE:** Do not overwrite existing files on the Destination Server. |
| **priorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY**. With this value, the Manager uses the default priority for this request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] 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:<br>• A null string to specify no options.<br>• **-login:** Login is used for some sources. This option obsoletes the **–gateway** option of the previous version.<br>• **-pass:** Password used in conjunction with the **–login** option for some sources. |

| Variable | Description |
|---|---|
| **requestNumber** | Number identifying the request. |

### Description

Restores an object from a specific instance. If this instance is externalized, the operation fails even if there are other instances available for the object.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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** | 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. |

| Value | Description |
|---|---|
| **DIVA_ERR_SEVERAL_OBJECTS** | More than one object with the specified name exists in the DIVArchive Database. |
| **DIVA_ERR_INSTANCE_OFFLINE** | Instance specified for restoring this object is ejected, or the Actor owning the specified Disk Instance is not available. |
| **DIVA_ERR_INSTANCE_DOESNT_EXIST** | Instance specified for restoring this object does not exist. |
| **DIVA_ERR_OBJECT_IN_USE** | The object is currently in use (*being Archived, Restored, Deleted, etc.*). |
| **DIVA_ERR_SOURCE** or **DESTINATION_DOESNT_EXIST** | The specified source 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**
- **DIVA_getObjectInfo**

### 2.9.33 DIVA_restoreObject

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_restoreObject (
IN DIVA_STRING              objectName,
IN DIVA_STRING              objectCategory,
IN DIVA_STRING              destination,
IN DIVA_STRING              filesPathRoot,
IN DIVA_RESTORE_QOS         qualityOfService,
IN int                      priorityLevel,
IN DIVA_STRING              restoreOptions,
OUT int                     *requestNumber
);
```

| Variable | Description |
|---|---|
| **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 (*this may result in an error if several objects have the same name*). |
| **Destination** | Destination (*e.g. video server or browsing server*) for the object files. This name must be known by the DIVArchive Configuration Description. |
| **filesPathRoot** | Root folder on the destination where the object files will be placed. If null (***string("")***), the files will be placed in the **FILES_PATH_ROOT** folder specified when archiving the object (*using the **DIVA_archiveObject()** function*). |

| Variable | Description |
|---|---|
| **qualityOfService** | One of the following codes:<br>**DIVA_QOS_DEFAULT**: Restoring is performed according to the default Quality Of Service (*currently: direct and cache for restore operations*).<br>**DIVA_QOS_CACHE_ONLY**: Use cache restore only.<br>**DIVA_QOS_DIRECT_ONLY**: Use direct restore only.<br>**DIVA_QOS_CACHE_AND_DIRECT**: Use cache restore if available or direct restore if cache restore is not available.<br>**DIVA_QOS_DIRECT_AND_CACHE**: Use direct restore if available or cache restore if direct restore is not available.<br>**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.<br>**DIVA_QOS_NEARLINE_AND_DIRECT**: Use Nearline restore if available, or direct restore if Nearline restore is not available.<br>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:<br>**DIVA_RESTORE_SERVICE_DO_NOT_OVERWRITE**: Do not overwrite existing files on the destination server.<br>**DIVA_RESTORE_SERVICE_DO_NOT_CHECK_EXISTENCE**: Do not check existence of the clip on the server.<br>**DIVA_RESTORE_SERVICE_DELETE_AND_WRITE**: Force delete and rewrite if object exists on the server.<br>**DIVA_RESTORE_SERVICE_DEFAULT**: Operate using the default setting in the Manager Configuration. |

| Variable | Description |
|---|---|
| **priorityLevel** | Level of priority for this Request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY**. With this value the Manager uses the default priority for this request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] 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 Database. Currently the possible values for **restoreOptions** are:<br>• A null string to specify no options.<br>• **-login**: Login is used for some sources. This option obsoletes the **-gateway** option of the previous version.<br>• **-pass**: Password used in conjunction with the **-login** option for some sources. |
| **requestNumber** | Request Number assigned to this request. This number is used for querying the status or cancelling this request |

## Description

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()**.

In the event 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.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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** | 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. |

| Value | Description |
|---|---|
| **DIVA_ERR_OBJECT_IN_USE** | The object is currently in use (*being Archived, Restored, Deleted, etc.*). |
| **DIVA_ERR_SOURCE** or **DESTINATION_DOESNT_EXIST** | The specified source 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**
- **DIVA_copyToGroup**

### 2.9.34 DIVA_transcodeArchive

#### Synopsis

```
#include "DIVAapi.h"

DIVA_STATUS DIVA_transcodeArchive (
DIVA_STRING                 parentObjectName,
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
);
```

| Variable | Description |
|---|---|
| **parentObjectName** | Name of the original object to be transcoded. |
| **parentObjectCategory** | Category of the original object. |
| **instance** | Instance of the Parent Object. The default is **-1**. |
| **objectName** | Name of the transcoded object which is the result of the transcoding operation. |
| **objectCategory** | Category of the transcoded object. |

| Variable | Description |
|---|---|
| **mediaName** | The Tape Group or Disk Array on which the object is to be saved. The media may be defined as follows:<br>1. **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*).<br>2. **SP Name** – Provide a Storage Plan Name as defined in the configuration. The object will be saved to the default media specified in the SP and assigned to the specified SP.<br>3. **Both 1 and 2: Name "&" SP Name** – The object is saved to the specified media as in number 1 above. The object is assigned to the specified SP as in number 2 above. The Media Name and the SP Name must be separated by the delimiter "&" (*configurable*).<br>When this parameter is a null string, the default group of tapes called **DEFAULT** is used.<br>Complex Objects may only be saved to AXF media types. |
| **comments** | Optional information describing the object (*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:<br>**-tr_archive_format FORMAT** (*required*)<br>Destination format of the retrieved object.<br>**-tr_names trans1** or<br>**-tr_names trans1,trans2** (*optional*)<br>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. |

| Variable | Description |
|---|---|
| **qualityOfService** | One of the following codes:<br>**DIVA_QOS_DEFAULT**: Archiving is performed according to the default Quality Of Service (*currently: cache only for Archive operations*).<br>**DIVA_QOS_CACHE_ONLY**: Use cache archive only.<br>**DIVA_QOS_DIRECT_ONLY**: Use direct archive only. No Disk Instance is created.<br>**DIVA_QOS_CACHE_AND_DIRECT**: Use cache archive if available or direct archive if cache archive is not available.<br>**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**, then both the original object and the transcoded object will be deleted. |
| **priorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY**. With this value, the Manager uses the default priority for this Request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] 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 cancelling this request |

### Description

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.

### Return Values

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

| Value | Description |
|---|---|
| **`DIVA_OK`** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **`DIVA_ERR_NOT_CONNECTED`** | No open connection. |
| **`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 has been broken. |
| **`DIVA_ERR_TIMEOUT`** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **`DIVA_API_TIMEOUT`** variable and equals 180 seconds by default. |
| **`DIVA_ERR_UNKNOWN`** | An unknown status has been received from the DIVArchive Manager. |
| **`DIVA_ERR_INTERNAL`** | An internal error has been 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`** | 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_ALREADY_EXISTS`** | An object with this name and category already exists in the DIVArchive System. |

| Value | Description |
|---|---|
| **DIVA_ERR_OBJECT_PARTIALLY_DELETED** | The specified object has instances that are partially deleted. |

**See Also:** **DIVA_linkObjects**

### *2.9.35 DIVA_transferFiles*

#### 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
);
```

| Variable | Description |
|---|---|
| **source** | Name of the Source (*e.g. video server, 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 Pathnames relative to the folder specified by the **sourcePathRoot** parameter. When the **sourcePathRoot** is null, pathnames must be absolute names. |
| **destination** | Name of the Destination (*e.g. video server, browsing server*). This name must be known by the DIVArchive Configuration Description. |
| **destinationPathRoot** | Root folder where the files will be placed at the destination. |

| Variable | Description |
|---|---|
| **priorityLevel** | Level of priority for this request. The **priorityLevel** can be in the range [0...100] or the value **DIVA_DEFAULT_REQUEST_PRIORITY**. The value 0 is the lowest priority and 100 the highest.<br>There are five predefined values:<br>• **DIVA_REQUEST_PRIORITY_MIN**<br>• **DIVA_REQUEST_PRIORITY_LOW**<br>• **DIVA_REQUEST_PRIORITY_NORMAL**<br>• **DIVA_REQUEST_PRIORITY_HIGH**<br>• **DIVA_REQUEST_PRIORITY_MAX**<br>Another predefined value is **DIVA_DEFAULT_REQUEST_PRIORITY**. With this value, the Manager uses the default priority for this Request (*default Request Priority is defined in the Manager Configuration*).<br>Using another value (*out of the range [0...100] 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 cancelling this request |

### Description

Submits a **Transfer Files Request** to the DIVArchive Manager. The request will transfer files from a remote server (*Source*) to another remote server (*Destination*). This function returns as soon as the Manager accepts the request. To check that the operation was completed successfully, the application must call the function **DIVA_getRequestInfo()**.

### Return Values

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

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **DIVA_ERR_SYSTEM_IDLE** | The DIVArchive System is no longer able to accept connections and queries. |

| Value | Description |
|---|---|
| **DIVA_ERR_BROKEN_CONNECTION** | The connection with the DIVArchive Manager has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been 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** | 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_SOURCE or DESTINATION_DOESNT_EXIST** | The specified Source/Destination is not known by the DIVArchive System. |

**See Also:** **DIVA_getRequestInfo**

### 2.9.36 DIVA_unlockObject

#### Synopsis

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

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

```
DIVA_STATUS DIVA_unlockObject (
IN DIVA_STRING        objectNme,
IN DIVA_STRING        category,
IN string             options
);
```

| Variable | Description |
|---|---|
| **objectName** | Name of the object. |
| **category** | The category to which the object was assigned when archived. |
| **options** | TBD |

#### Return Values

| Value | Description |
|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. |
| **DIVA_ERR_NOT_CONNECTED** | No open connection. |
| **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 has been broken. |
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by the DIVArchive API. |

# 3 Using the DIVArchive API with DIVAnet

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

## *3.1 What is DIVAnet?*

DIVAnet provides a unified view of archived content across multiple, distributed DIVArchive systems. It facilitates the moving of content back and forth among DIVArchive sites, as well as 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, such as restores and copies. DIVAnet also provides access rules to limit the operations that users are allowed to perform.

## *3.2 API Support*

DIVAnet has partial support for the full DIVArchive API command set. Refer to the appropriate DIVAnet Guide for a complete list of supported API commands. DIVAnet will support client connections from DIVArchive API clients versioned 7.3 and earlier. New parameters or features added to the API after release 7.3 are not supported by DIVAnet. In general, a released version of DIVAnet has the ability to connect to newer releases of DIVArchive, and sometimes has the ability to connect to older releases as well. This ability varies based on the specific release of DIVAnet. Refer to the appropriate DIVAnet Guide for more information on DIVArchive API support.

## *3.3 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 the well-known DIVArchive API parameters in a slightly a different way.

For example, the DIVAnet Copy command (*CopyToGroup*) can be used to copy 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. One method is to prefix the **`target_sitename`** to the media provided in the call (*for example,* ***`sitename2_mytapegroup`***). Refer to the appropriate DIVAnet Guide for more information on how to specify DIVAnet-specific information in DIVArchive API calls.

### 3.4 Returned Parameters

A DIVAnet system will sometimes return API information that is slightly different than you would typically see in a DIVArchive system. For instance, the DIVAnet **`getObjectInfo()`** call returns information about an archived object across all DIVArchive sites. In order to distinguish which site is which, the source sitename 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 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.

Note also that the Request ID returned by each DIVAnet request does not necessarily correspond to a DIVArchive request ID – refer to the appropriate DIVAnet Guide for more information.

### 3.5 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 may possibly refuse an API connection altogether because of configured permissions (*whereas DIVArchive will accept the connection - if it hasn't run out of available connections*). Finally, there are cases where DIVAnet will choose to acknowledge a request with **`DIVA_OK`** and then later return an error (*for example, an Invalid Media error*), where DIVArchive will simply reject the request with the **`DIVA_ERR_INVALID_PARAMETER`** error.

### 3.6 GetObjectDetailsList call

The **`GetObjectDetailsList()`** command is used to retrieve 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 may receive the same object again in the next batch (*for the second site*). This makes **`GetObjectDetailsList()`** different from **`GetObjectInfo()`** (***`GetObjectInfo()`*** *returns information from all sites in one call*).

The query parameters and time ranges queried in each batch are specific to each site. This means that it is possible that if Site1 contains many objects in a given query (*and Site2 does not*), Site2's batches (*towards the end of the calling sequence*) may be completely empty.

Keep calling **GetObjectDetailsList()**, ignoring empty batches until the call returns a status of **DIVA_WARN_NO_MORE_OBJECTS**, or an error. All DIVArchive sites in the DIVAnet network must be online in order for **GetObjectDetailsList()** to succeed. If an error is returned before the list has been fully returned (*for any reason*), 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 the **MODIFIED_SINCE** parameter is used. 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 so, provide the exact same query information (*passing all received the 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.

For more information on specific DIVArchive API calls, refer to the appropriate DIVAnet Guide.

# APPENDIX

## *A1 List of Special Authorized Characters in DIVArchive*

| Characters / Fields | Name | Category | Source | Media | Path | File | Comments | Options |
|---|---|---|---|---|---|---|---|---|
| ~ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ` | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ! | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| @ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| # | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| % | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ^ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| & | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | NO |
| * | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |
| ( | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| _ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| + | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| = | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
\| | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |
| \ | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |
| } | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| { | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| [ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

| Characters / Fields | Name | Category | Source | Media | Path | File | Comments | Options |
|---|---|---|---|---|---|---|---|---|
| : | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |
| ; | ✓ | ✓ | ✓ | ✓ | ✓(1) | ✓ | ✓ | ✓ |
| “ | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | NO |
| ‘ | ✓ | ✓ | NO | NO | ✓(1) | ✓ | ✓ | ✓ |
| < | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | NO |
| , | ✓ | ✓ | ✓ | ✓ | ✓(1) | ✓ | ✓ | ✓ |
| > | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |
| . | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |
| ? | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |
| / | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |
| Space | ✓ | ✓ | ✓ | ✓ | NO | ✓ | ✓ | ✓ |

**Note: In a Windows environment, the following File/Folder Name restrictions apply:**

***(1)* Depends upon file system restrictions.**

**File/Folder names cannot consist solely of one or more space(*s*).**

**File/Folder names cannot contain a double-quote (i.e. “).**

### *A2 Maximum number characters allowed*

| | Name | Category | Source | Media | Path and Filename (*per file or per folder*) | Comments | Options |
|---|---|---|---|---|---|---|---|
| **Maximum number of characters** | 192 | 96 | 96 | 96 | 1536 | 4000 | 768 |

## A3 API Static Constants

| Static Constant Name | Description | Values |
|---|---|---|
| **DIVA_OK** | The request has been correctly submitted and accepted by the DIVArchive Manager. | 1000 |
| **DIVA_ERR_UNKNOWN** | An unknown status has been received from the DIVArchive Manager. | 1001 |
| **DIVA_ERR_INTERNAL** | An internal error has been detected by the DIVArchive Manager or by 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 has been 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 version of the API and the Manager are not compatible. | 1007 |
| **DIVA_ERR_INVALID_PARAMETER** | A parameter value has not been 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** | **requestNumber** identifies no request. | 1011 |

| Static Constant Name | Description | Values |
|---|---|---|
| **DIVA_ERR_NOT_CANCELABLE** | The request is at the point where it is not cancellable. | 1012 |
| **DIVA_ERR_SYSTEM_IDLE** | The DIVArchive System is no longer able to accept connections and queries. | 1013 |
| **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 has not been 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 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 has been 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 |

| Static Constant Name | Description | Values |
|---|---|---|
| **DIVA_ERR_TIMEOUT** | Timeout limit has been reached before communication with the DIVArchive Manager could be performed.<br>Timeout duration is set by the **DIVA_API_TIMEOUT** variable and equals 180 seconds by default. | 1024 |
| **DIVA_ERR_LAST_INSTANCE** | **DIVA_deleteObject()** must be used to delete the last instance of an object. | 1025 |
| **DIVA_ERR_PATH_DESTINATION** | The specified Destination Path is invalid. | 1026 |
| **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 not available. | 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, etc.*). | 1031 |
| **DIVA_ERR_CANNOT_ACCEPT_MORE_REQUESTS** | Count of simultaneous requests reached the maximum allowed value. This variable is set in the manager.conf configuration file. The default is 300. | 1032 |
| **DIVA_ERR_TAPE_DOESNT_EXIST** | There is no tape associated with the given barcode. | 1033 |

| Static Constant Name | Description | Values |
|---|---|---|
| **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 be locked. A locked object cannot be Restored or Copied to New. | 1039 |
| **DIVA_ALL_REQUESTS** | Specify all requests. Used by **DIVA_cancelRequest.** | -2 |
| **DIVA_ALL_INSTANCE** | Speficy all instances. Used by **DIVA_release.** | -1 |
| **DIVA_ANY_INSTANCE** | **Allow Manager to choose the instance.** | -1 |
| **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 |

| Static Constant Name | Description | Values |
|---|---|---|
| **DIVA_MEDIA_FORMAT_LEGACY** | The specified media format for the group or array is Legacy. | 0 |
| **DIVA_MEDIA_FORMAT_AXF** | The specified media format for the group or array is the 0.9 release of AXF. | 1 |
| **DIVA_MEDIA_FORMAT_AXF_10** | The specified media format for the group or array is the 1.0 release of AXF. | 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 |