Oracle® DIVArchive

Checksum Support User's Guide

Release 7.4

E73118-01

June 2016

Oracle DIVArchive Checksum Support User's Guide, Release 7.4

E73118-01

Primary Author: Lou Bonaventura

Contents

Preface

This document introduces the Checksum Support and Content Verification program for the Oracle DIVArchive Suite.

Audience

This document guides System Administrators through checksum configuration, and users through checksum verification, operational processes, and functionality.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information on Oracle DIVArchive refer to the additional DIVArchive documentation libraries.

Conventions

The following text conventions are used in this document:

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

1 Introduction

The Checksum Support and Content Verification program introduces additional levels of data verification into the Oracle DIVArchive Suite. This feature brings checksum generation and verification for each file and object managed by DIVArchive.

Oracle Linux 7 (x86_64, 64-bit) is now supported for all core DIVArchive components. If you require a Linux environment in a language other than English, simply create a user and identify the desired language in the user profile. Oracle Linux has support for a variety of languages (other than English) and the language can be selected during Linux installation. Windows installations must use the English language and Oracle only supports English-based Windows environments.

All Windows batch files (`.bat`) have corresponding shell scripts (`.sh`) in Linux. You must substitute Windows paths with Linux paths when operating on Linux. For example, the Windows path `C:\DIVA\Program` will be `/home/diva/DIVA/Program` when running under Linux.

Note: Linux commands, paths and file names are case-sensitive.

See the *Oracle DIVArchive Supported Environments Guide* in the *Oracle DIVArchive 7.4 Core Documentation Library* for information about certain limitations when running in the Linux environment.

Supported Checksum Algorithms and Sources

The available checksum algorithms for Oracle DIVArchive are as follows:

Checksum Algorithm MD2
Message Digest 2 (MD2) Algorithm: A cryptographic hash function developed in 1989. The algorithm is optimized for 8-bit computers. Although other algorithms have been proposed since, MD2 remains in use in public key infrastructures as part of certificates generated with MD2 and RSA.

Checksum Algorithm MDC2
Modification Detection Code 2: In cryptography MDC2 (sometimes called Meyer-Schilling) is a cryptographic hash function with a 128-bit hash value. MDC2 is based on a block cipher with a proof of security in the ideal-cipher model.

Checksum Algorithm MD5
Message Digest 5 Algorithm: In cryptography, MD5 is a widely used cryptographic hash function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications and is commonly used to check the integrity of files. *MD5 is the default DIVArchive Checksum Type.*

Checksum Algorithm SHA

Secure Hash Algorithm: One of several cryptographic hash functions published by the National Institute of Standards and Technology as a US Federal Information Processing Standard.

Checksum Algorithm SHA-1

Secure Hash Algorithm-1: A 160-bit hash function which resembles the MD5 algorithm. This was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm. *SHA-1 is the default SAMMAsolo Checksum Type.*

Checksum Algorithm RIPEMD160

RACE Integrity Primitives Evaluation Message Digest: A 160-bit message digest algorithm (cryptographic hash function) developed in Leuven (Belgium) at the COSIC Research Group at the Katholieke Universiteit Leuven, and first published in 1996. It is an improved version of RIPEMD, which was based upon the design principles used in MD4, and is similar in performance to the more popular SHA-1.

When an object contains multiple files (components), a checksum will be generated and later verified for each of the component elements. There are three types of checksums that are distinguishable:

Genuine Checksum (GC)

This checksum is provided through the API in the Archive request, or retrieved by the Oracle DIVArchive Actor from the Source/Destination. It ensures maximum security because it allows DIVArchive to verify all transfers to and within the archive system.

The GC is obtained before the archive starts. It is either passed in the *archiveObject* API function, or obtained from the Source/Destination by the Oracle DIVArchive Actor using an API provided by the Source/Destination manufacturer.

Archive Checksum (AC)

This checksum is generated during the transfer phase into DIVArchive and based on the data that is received from the network (for networked sources), calculated during the actual transfer, or read from the device (for disk type sources). This type of checksum does not allow detecting corruptions occurring during the transfer from the Source/Destination to the Actor. However, all other subsequent corruptions can be detected.

The AC is calculated real-time before it writing to disk or other storage medium (within DIVArchive) during data transfer.

Deferred Checksum (DC)

This checksum is generated during the read of an object already stored in the archive system which has no checksum previously associated with it - either because the previous DIVArchive version did not support it, or the option was not activated.

This type of checksum does not allow corruption detection that occurred during an earlier stage (for example, during the archive or further data movement within a copy or repack process). However, it enables corruption detection in all further data processing.

This checksum is generated during requests on existing objects. (for example, Copy, Restore, and so on).

Checksum Verification Workflows and Supported Requests

Checksum verification occurs when transferring data from a Source/Destination or when reading data from a source or a storage medium. The latter occurs during the retrieval of an object from a storage medium during routine functions (Restore, Copy, Repack, and Transcode, but not Oracle DIVArchive Partial File Restore), during a read-back from storage (Verify Following Write feature), or from the source (Verify Following Restore feature). MD5 is the default DIVArchive Checksum Type which applies to the entire system.

There are four major workflows used for Checksum Support Verification. Each of these workflows are described as follows:

Verify Read (VR)
Verify Read is the default workflow. It calculates a real-time checksum for content as it is being read from a storage medium (for example, a disk or a tape - including cache disks configured for VR) inside DIVArchive, and performs checksum verification. The operation is only considered successful after this full read operation is complete and the calculated checksum matches the checksum attached to the stored data.

Verify Write (VW)
Verify Write reads back data that was just written to a storage medium, for example a disk or a tape inside DIVArchive, and performs checksum verification. This read-back data will be used to calculate a real-time checksum and then discarded. The write operation is successful when the full read operation is complete and the calculated checksum matches the checksum of the incoming data.

Verify Following Archive (VFA)
Verify Following Archive re-transfers the data from the source device after an initial archive operation, and then compares it against the AC checksum calculated. The archive operation is successful when the second transfer is fully complete and the checksums are identical. ***This verification mode is not compatible with GC enabled sources or complex objects.***

Verify Following Restore (VFR)
Verify Following Restore re-transfers the data from the source device after an initial restore operation, and then compares it with the checksum attached to the stored data. This restore operation is considered successful when the second transfer is fully complete and the checksums are identical.

WARNING: This verification mode is not compatible with complex objects.

Each workflow can be used with one or several requests. The following list describes the DIVArchive requests and the appropriate corresponding checksum workflows. The operation is listed and then the supported checksum workflows for that operation.

Notes: *Verify Following Archive* and *Genuine Checksum* are mutually exclusive.

Verify Write does not apply to cache disk writes.

Archive
Verify Read (default), Genuine Checksum, Verify Following Archive, Verify Write, SAMMAsolo integration

Restore
Verify Read (default), Verify Following Restore

N-Restore
Verify Read (default)

Partial File Restore
Checksums are not supported for Partial File Restore.

Copy
Verify Read (default), Verify Write

Copy As New
Verify Read (default), Verify Write

Associative Copy
Verify Read (default), Verify Write

Verify Tapes
Verify Read (default)

Repack Tapes
Verify Read (default), Verify Write

Export
Export content with checksum (default)

Import
Import content with checksum (default)

Transcoding (Archive, Restore, Copy)
Verify Read, Genuine Checksum - both workflows are supported with a change in the object format.

Checksum Support with Complex Objects

DIVArchive 7.4 supports all checksum workflows for non-complex objects. However, only Verify Write (VW) is supported for complex objects.

Complex object checksums are stored in the Metadata database rather than the Oracle database, and therefore will not be displayed in any database queries. The `getObjectInfo` API call will return a phony checksum, and not all files and folders will be displayed (only a single file representing the entire complex object).

If checksum support is disabled when a complex object is archived, and then subsequently enabled, there will be no checksum comparison during operations on the complex object. In other words, whatever checksum is used when the complex object is archived will be the checksum used throughout the life of the object.

2 Checksum Support Configuration

You configure checksum support through the DIVArchive Configuration Utility using the *Engineer* login. The following sections describe each option and how to adjust the settings.

Global Checksum Parameters

You access the global checksum parameters under the **Manager Settings** tab in the Configuration Utility using the *Engineer* login. Each of the global parameters affects all checksum support settings throughout the system. The following are the global parameter descriptions.

Manager: Default Checksum type
There are several checksum algorithms available on the system including MD2, MD5, SHA, SHA-1, MDC2, and RIPEMD160. The MD5 algorithm is the default checksum for the DIVArchive system. To change the ***Default Checksum Type*** in use, select the desired default checksum type from the Default Checksum Type list.

Manager: Number of retries following failed checksum
This parameter sets the number of times the system will retry the operation after a failed checksum. The default (and recommended) setting is one retry after a checksum failure. Enter the number of retries allowable for your data and system in the ***Number of retries following failed checksum*** field.

Note: A checksum error should be an extremely rare event, and typically indicate an issue with the hardware or the network.

Manager: Select different drive per retry on failed checksum
This parameter distinguishes whether the retry (after a failed checksum) will be attempted on the same drive (option deselected) or if the system should try the operation using a different drive (option selected). The default setting for this parameter uses the same drive (deselected).

Source/Destination Checksum Configuration

You change the checksum support configuration for Source/Destinations on the **System** tab in the Configuration Utility using the following procedure:

1. Navigate to the **System** tab in the Configuration Utility.
2. Double-click the ***Source*** or ***Destination*** for which checksum configuration is required (on the right side of the screen).

3. The Edit Sources and Destinations Entry dialog box appears with several checksum support options. These options are mainly associated with the Genuine Checksum type.

   The ***Genuine Checksum Source*** must be used (**Yes**) for the system to read the checksum from the external source providing the file. This initiates a real-time checksum calculation to compare the checksums and verify the initial transfer.

4. Select the desired ***Checksum Type*** using the provided list - all supported checksum types are listed.

   The Genuine Checksum is only used for the first verification. The selected ***Checksum Type*** is only used once and then discarded. Beyond the initial use of the selected Checksum Type (after the transfer), the default type is used. The ***Checksum Type*** and ***GC Mode*** must match the settings implemented at the source.

5. Select the desired ***Genuine Checksum Mode*** from the list. This selection tells the Actor the format of the files containing the checksum data. Available options are **MDF_XML**, **TEXT** and **AXF**.

   - When ***Verify Following Archive (VFA)*** is turned on (selected), performing the initial transfer from the source results in a read-back operation and therefore the data is being read twice for verification. After the data is read twice, the two checksums are compared. If they are the same then verification is successful, if not identical then verification fails.

     Verify Following Archive is *not* compatible with Genuine Checksum or complex objects. There is no need to use VFA when GC is being used because the checksum is already verified. The Genuine Checksum must be turned off to gain access to the VFA check box. If GC is turned on, the VFA check box will be grayed out and not selectable.

   - When ***Verify Following Restore (VFR)*** is turned on (selected), performing the final transfer to the destination results in a read-back operation where the data is being read twice for verification. A checksum is calculated during the read-back and compared to the checksum attached to the stored data. Verification is successful if they match, if they are not identical then verification fails. The setting of GC has no bearing on the VFR setting.

     Note: ***Verify Following Restore*** is not compatible with complex objects or the **-axf** option.

     Verify Following Restore was designed to read back the restored content from a video server to confirm it is not corrupt. Using the **-axf** option does not create a checksum-verifiable restore. It creates an object export that is encompassed in an AXF Wrapper (container). VFR and **-axf** are mutually exclusive and should not be part of the same workflow.

6. Click **OK** to complete the process.

Array and Disk Checksum Configuration

The Verify Write (VW) functionality can be turned on or off either on an array basis, or disk by disk. VW applies when you write to the final storage location in DIVArchive. When turned on, the system will perform a read-back of what was just written and compare the checksum calculated during the read-back to the checksum attached to the incoming data.

The Verify Write (VW) column indicates whether the ***Verify Write*** function is on or off for the particular array and disk. The default setting is off. If there is nothing defined in the Disk VW column, the system will use the setting defined in the Array VW column.

Use the following procedure to change the Checksum Support configuration in the Configuration Utility to override the setting defined in the Array VW column for a specific disk:

1. Navigate to the **Disks** tab in the Configuration Utility.
2. Select the disk to configure, and then click **Edit** located just above the Disk VW column. The Edit Disks Entry dialog box appears.
3. When the dialog box appears, use the ***Verify Write*** list to select **ON**, **OFF**, or **NONE** (the blank selection). If **NONE** is selected, the ***Verify Write*** will use the setting identified in the array for this particular disk.
4. Click **OK** to complete the process.

The selection made in the Edit Disks Entry dialog box is reflected in the Disks VW column.

Group Checksum Configuration

You configure Verify Write (configurable by Groups) in the VW column in the Configuration Utility using the following procedure:

1. Navigate to the Groups area of the **Sets, Groups & Media Mapping** tab in the Configuration Utility.
2. Select the Group to configure.
3. Click **Edit** and select **ON** or **OFF** using the ***Verify Write*** list.
4. Click **OK** to complete the process.

The selection made will be reflected in the VW column on the Groups display area. When writing a file to a particular group, the setting for that group will be applied to the file. The default setting for Groups is off.

Actor Checksum Configuration

This setting defines if the Actor is automatically selected for the Verify Tape workflow. All Actors are included by default, however you can exclude specific Actors as necessary. You configure Verify Tape for Actors in the Configuration Utility using the following procedure:

1. Navigate to the Actors area under the **System** tab.
2. Select the Actor to configure.
3. Click **Edit** and select **Yes (Y)** or **No (N)** using the ***Verify Tape*** list.
4. Click **OK** to complete the process.

Oracle SAMMAsolo Integration Configuration

Genuine Checksums are passed from SAMMA to DIVArchive using an XML Metadata file. When the Actor is instructed to read a file using an Source/Destination configured for GC, the Actor generates checksums according to the configuration. These values, and the values obtained from the metadata file, will be transferred to the Manager for comparison.

For example, for a file named `sample.abc`, the metadata file name and structure are in an XML-formatted file with the following parameters (as a minimum):

```
<DIVAObjectDefinition>
  <objectName>sample</objectName>
  <fileList>
    <file checksumType="SHA1"
       checksumValue="9F097AAEEF48C4A170D95AAF6161790662626802">
       sample.abc
    </file>
    </fileList>
</DIVAObjectDefinition>
```

The resulting metadata file name will be `sample.abc.xml`.

SAMMAsolo Configuration Process

There are three aspects to the SAMMAsolo checksum configuration as follows:

Note: The configuration procedures described below are samples provided for your convenience. Actual procedures may vary depending on your software release. Refer to the applicable product documentation for more details.

Step1: SAMMAsolo Configuration

Use the following procedure to configure SAMMAsolo to drop the migrated clip and XML file into a specific folder:

1. Start SAMMAsolo.
2. Click **File**, and then click **New**.
3. Open an sxt template (sxt template files are located in the `C:\Program Files\Front Porch Digital\SAMMA Solo` folder).
4. Click **Configuration**, **Administrator**, and then **Login** to log in as an Administrator.
5. Click **Configuration**, and then **Encoders**.
6. Navigate through each encoder tab and change the setting for ***Move File to Here on Successful Migration***. This is where the migrated clip and XML metadata file will be dropped by SAMMAsolo (for example `d:\dfm_folder`).
7. Save the sxt template for future use by clicking **File**, and then **Save**.
8. Place the `divaXML.vbs` file in the folder `C:\Program Files\Front Porch Digital\Common`. This file is provided by SAMMAsolo.
9. Click **Configuration**, and then click **Options**.
10. Click the **Commands** tab and set the following command for ***On Success***:
    - `C:\WINDOWS\system32\cscript.exe`
    - `C:\Program Files\Front Porch Digital\Common\divaXML.vbs`
    - `F:\Success\$(Settings\Details\Filename).xml`
11. Configure FTP on the SAMMAsolo system and add the `d:\dfm_folder` folder so it is accessible through FTP.

Step2: DFM Configuration

You must install and preconfigure DFM according to the *Oracle DIVArchive DFM User's Guide*. Use the following procedure to configure DFM to monitor the specific folder on the SAMMAsolo system.

When DFM is used in a Linux environment to monitor an FTP folder, it must be configured as in the following example:

User: `diva`

User Home Directory: `/ifs`

Folder to be Monitored: `/ifs/folder1`

Correct DFM Configuration: `ftp://diva:password@host_ip/folder1`

Incorrect DFM Configuration: `ftp://diva:password@host_ip/ifs/folder1`

In the DFM configuration file, configure a Single type folder to point to the SAMMAsolo System.

```
<folderConfig>
  <!-- Folder URL. -->
  <url>ftp://diva:diva@172.16.3.47/</url>
  <type>single</type>
  <priority>30</priority>
  <mdfConfigPriority>Primary</mdfConfigPriority>
  <categoryName>dfm_solo</categoryName>
  <incompleteThreshold>86400</incompleteThreshold>
  <sourceDestinationDIVAName>dfm_solo</sourceDestinationDIVAName>
  <archiveFilePathTemplate platform="DETECT"
   options="">URL_TO_FILE</archiveFilePathTemplate>
  <archiveFileNameTemplate platform="DETECT"
   options="">filename.ext</archiveFileNameTemplate>
  <divaMediaName>array_01</divaMediaName>
  <fileFilter type="exclude">
  <mask>*.XML</mask>
  </fileFilter>
  <deleteBeforeArchive>TRUE</deleteBeforeArchive>
</folderConfig>
```

The following significant parameters in the above example are described as:

`<url>`
The FTP connection details for the SAMMAsolo system.

`<type>`
The type of folder DFM is monitoring (*Single* or *File Set*).

`<SourceDestinationDIVAName>`
The name of the source used when DFM submits the archive request to the Manager.

`<divaMediaName>`
The name of the media that DFM will submit the archive request to.

`<fileFilter>`
Prevents DFM from archiving the metadata XML file.

Step 3: DIVArchive Configuration

Use the following procedure to configure DIVArchive:

1. Add the Source/Destination declared in the DFM configuration file.
2. Add the Storage Media Name declared in the DFM configuration file.
3. The source must be pointing to the SAMMAsolo system FTP.
4. The source must be configured with the following options:
   - ***Checksum Type*** must be set to **SHA1**
   - ***GC_Mode*** must be set to **MDF_XML**

3 Checksum Support Operations

In this chapter, you will examine checksum support operations to verify that the data you end up with matches the original data.

Verify Read (VR) Workflow (Default)

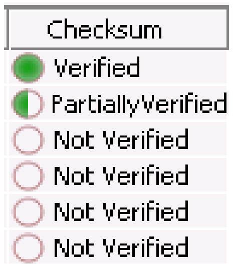
During the archive process, the checksum is generated real-time by the Actor and stored in the database. This checksum is not verified until an initial read-back or Restore operation is performed.

You view checksum verifications and failures through the Control GUI **Manage** tab in the Archived Objects view. The Archived Objects view shows the Checksum column, indicating the status of the checksum for any particular object in the system. The status is identified by circles (empty, partially filled, or fully filled) and text.

Double-clicking on the resource opens the Object Properties dialog box showing verification or failure messages and checksum information.

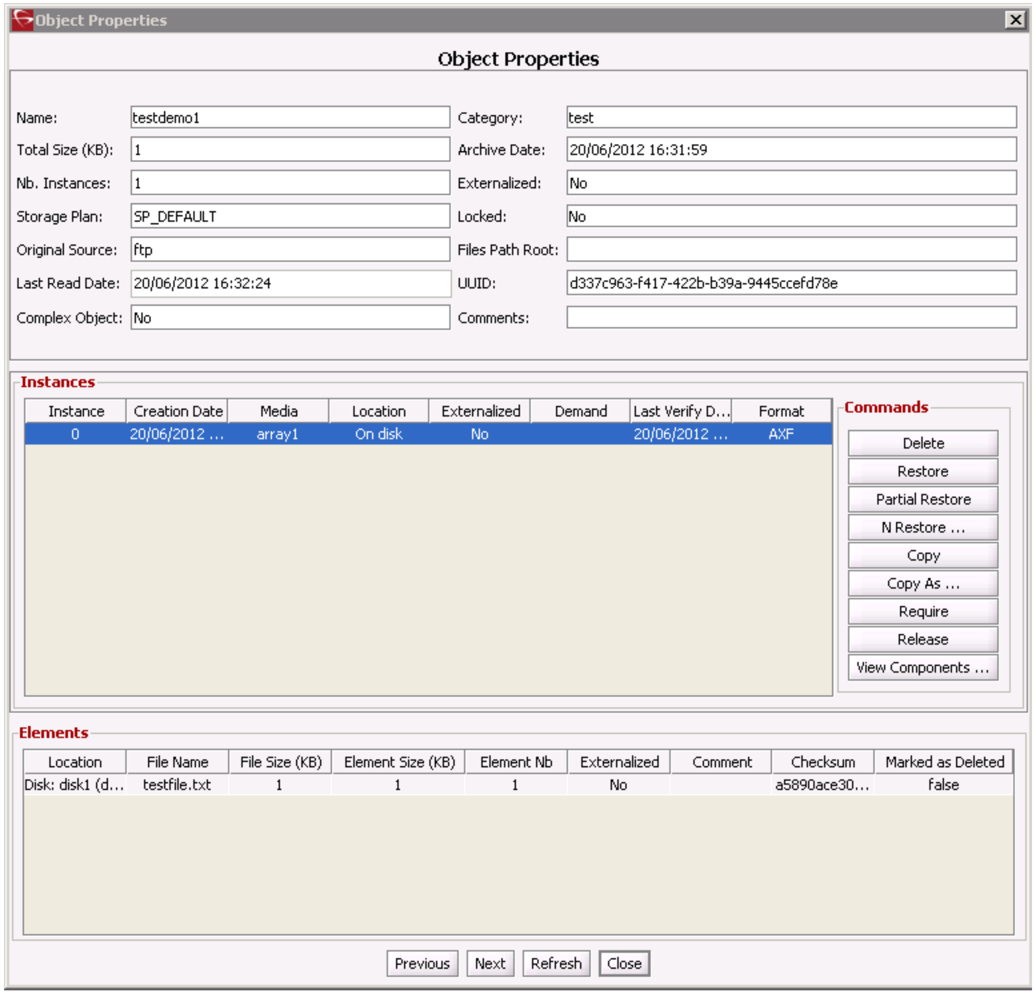
The Checksum column displays the following different status values:

- Not Verified (empty circle)
  - Verification has not been completed for this object.
    - The checksum has never been calculated because the object was archived in a software release before Oracle DIVArchive 6.4, or because it is a complex object.
    - The default checksum was used and the object has not been read-back for verification.
- Partially Verified (half green and half empty circle)
  - For objects with multiple instances this status appears if both of the following statements are true:
    - Verification succeeded for at least one instance.
    - Verification did not succeed, or has not been performed, for at least one instance.
- Verified (filled green circle)
  - Verification completed successfully.



Checksum
● Verified
◐ PartiallyVerified
○ Not Verified
○ Not Verified
○ Not Verified
○ Not Verified

In the following figure, although the highlighted object has not been verified, a checksum for that object was saved in the database. The Last Verify Date is viewable in the center portion of the screen labeled Instances. The checksum entered into the database is viewable in the bottom portion of the screen labeled Elements.



Object Properties

Name:	testdemo1	Category:	test
Total Size (KB):	1	Archive Date:	20/06/2012 16:31:59
Nb. Instances:	1	Externalized:	No
Storage Plan:	SP_DEFAULT	Locked:	No
Original Source:	ftp	Files Path Root:	
Last Read Date:	20/06/2012 16:32:24	UUID:	d337c963-f417-422b-b39a-9445ccefd78e
Complex Object:	No	Comments:	

Instances

Instance	Creation Date	Media	Location	Externalized	Demand	Last Verify D...	Format
0	20/06/2012 ...	array1	On disk	No		20/06/2012 ...	AXF

Elements

Location	File Name	File Size (KB)	Element Size (KB)	Element Nb	Externalized	Comment	Checksum	Marked as Deleted
Disk: disk1 (d...	testfile.txt	1	1	1	No		a5890ace30...	false

Checksum Failure Recording

Checksum failure dates are recorded in the DIVArchive database, however they are not (currently) displayed in the Control GUI. Whenever there is a checksum failure for a particular instance, the timestamp is stored for the instance in the database's `Instance` table. If a checksum failure occurs for an instance, the corresponding object's checksum status will be updated.

Example:

An object has two instances and the current object checksum status is **Fully Verified**. If a checksum failure occurs for one of the instances, the timestamp is recorded and the object status will be updated to **Partially Verified**.

- Instance A is on Tape-1 and verified.
- A restore of Instance A is made, however the checksum verification fails.
- The time of the failure is recorded in the DIVArchive database and the Checksum Status in the Control GUI is updated to **Partially Verified**.
  - This is because there is only one fully verified instance and one instance that failed verification.
  - The first instance of the object remains verified.

The following DIVAprotect daily metrics for checksum failures are available:

`TAPE_CHECKSUM_FAILURE_COUNT_DAY`
Tape operations checksum failure count.

`DISK_CHECKSUM_FAILURE_COUNT_DAY`
Disk operations checksum failure count.

`SD_CHECKSUM_FAILURE_COUNT_DAY`
Source/Destination operations checksum failure count.

Verify Following Archive (VFA) Workflow

Verify Following Archive (VFA) re-transfers the data from the source device after an initial archive operation and compares it against the calculated AC. Only after this full second transfer is completed, and the checksums compared, is the archive operation considered successful.

Note: This verification mode is not supported with GC-enabled sources or complex objects.

VFA reassures that there was no corruption introduced by the source gateway or network path to DIVArchive to the best level possible without a GC being passed. Generally, this verification will trap random errors introduced during the archive transfer into DIVArchive. However it will not discover more common corruptions (for example, header corruptions) introduced by a bug in the video server gateway. In this case the checksums will match, and the header corruption will not be detected.

Note: The impact on the video server (or gateway) performance and overall network bandwidth can be significant in this mode of operation.

Click the **Home** tab in the Control GUI, and then click the **Manager** icon to display the current Manager requests. Alternatively, you can click the **Manage** tab and then click the **Requests** icon to display the same view.

In either location, double-clicking the desired object opens the Request Properties dialog box, where you view the verification status of the checksum for that object.

These are the different verification status IDs you will see in the Events List display, and the overall VFA workflow for a successfully verified checksum:

- `ID 1143`: The system begins the archive process.
- `ID 1144`: The checksum is read from the source file.
- `ID 1145`: The VFA process starts.
- `ID 1147`: The checksum is compared to the original and correctly matches the original (from `ID 1143`) in the database.
- `ID 1148`: The read-back process begins.
- `ID 1150`: The checksum is compared to the original and correctly matches the original (from `ID 1143`) in the database.
- `ID 1152`: The verification succeeded because the checksum returned by the Actor matches the checksum value saved in the database.
- `ID 1153`: The request status is changed to **Completed**.

Verify Write (VW) Workflow

Verify Write (VW) reads back data that was just written to a storage medium (for example, a disk or tape) inside DIVArchive, and performs checksum verification. A real-time checksum is calculated using the read-back data, which is then discarded. The write operation is only considered successful after the full read operation is complete and the checksums are compared and verified to be identical.

The purpose of VW is to perform a read-after-write operation to compare the original checksum for the object elements with those calculated during the read-back operation. This guarantees no corruption was introduced because of disk, tape, or file system errors. VW is not required on cache disks since the subsequent read operation will trap any potential issues.

Note: The impact on DIVArchive disk bandwidth, internal network bandwidth, and (most importantly) on data tape operations are significant in this mode of operation.

Failed checksum verifications are indicated with red highlight in the Events List area in the Request Properties dialog box.

These are the different verification status IDs you will see in the Events List display, and the overall VW workflow for a successfully verified checksum:

- `ID 1146`: Displays the original checksum.
- `ID 1148`: The file is written to the tape and a read-back is initiated.
- `ID 1150`: The transfer is verified because the checksums match.
- `ID 1151`: The object is saved.
- `ID 1152`: This notification states that the instance has been verified using the checksum.

Verify Following Restore (VFR) Workflow

Verify Following Restore (VFR) re-transfers the data from the destination device after restoring, and then performs checksum verification. An on-the-fly checksum is calculated using the read-back data, which is then discarded. Only after the full second transfer is completed and the checksums compared is the archive operation considered successful. *This verification mode is not supported for complex objects.*

Note: This verification mode is not supported for complex objects.

After GC passes verification, VFR provides confidence that there was no corruption introduced by the destination gateway or network path from DIVArchive. This mechanism guarantees a full path restore verification since the restored item is fully transferred back to DIVArchive to calculate and compare a checksum value.

It is possible that some sources will not pass this verification check because they modify the restored files. For example, some video servers (for example, Leitch servers) will modify some headers upon restore. These sources should not be configured with VFR. The *Oracle DIVArchive Supported Environments Guide* document includes information on the compatibility of this feature with each specific Source/Destination device type.

Note: The impact on the video server or gateway performance, and overall bandwidth, can be significant in this mode of operation.

These are the different verification status IDs you viewable in the Events List display, and the overall VFR workflow for a successfully verified checksum:

- `ID 1120`: The VFR process is started and a second transfer is initiated
- `ID 1121`: The original checksum
- `ID 1122`: The transfer is verified because the checksums match
- `ID 1123`: VFR was completed successfully
- `ID 1124`: The request status is changed to **Completed**
- `ID 1125`: The instance has been verified (following restore) using the checksum

Verify Tape Request Workflow

Click the Tapes icon on the Control GUI's **Home** tab to display the Tapes screen. Double-clicking on one of the tapes, or on a Tape Group, results in the Tape Properties dialog box being displayed. In the Tape Properties dialog box there are columns for both the Checksum Value and the Verification Status for each component on the tape.

Use the following procedure to verify a tape:

1. Right-click the tape name that requires verification in the list shown on the Tapes screen.
2. Select the **Verify Tape** menu item from the resulting context menu - the Verify Tape dialog box is displayed
3. Select the request's ***Priority*** in the Verify Tapes dialog box.
4. Click **Send** to initialize the verification process.

During the verification process, the system will read-back through every object on the tape one at a time and verify all of the checksum values. If the checksum verification fails for a particular object, the verification process continues to the next object. The process continues running until the checksums of all objects have been checked (regardless of whether they failed).

Failed object checksum verification errors are displayed for that tape and indicated by red highlight in the Request Properties area on either the Manager screen in the **Home** tab, or the Requests screen in the **Manage** tab. The error will show the reason for the failure (checksums do not match), and the component that failed the verification.

If the verification of an object on the tape fails, the Logged Requests screen shows a status of **Partially Aborted** in the Status column. This indicates that the tape verification process examined everything on the tape, but there was at least one object that could not be verified. The checksum verification status is also displayed in the Tape Properties dialog box.

See Appendix A for information on tape verification limitations.

Tracking Checksum Errors in the DIVAprotect Journal

Checksum error events (generated by failed checksum verification) are displayed in the DIVAprotect Journal and indicated by red highlighting. There are three checksum events (see the following first table) that could be displayed, each having up to sixteen fields (see the following second table).

Note: Refer to the *Oracle DIVArchive DIVAprotect User's Guide* for detailed information.

The following table describes the checksum events:

Event ID	Event Name	Event Description	Severity
180	`CHECKSUM_ERROR_TAPE`	Checksum verification error while reading from tape.	2
181	`CHECKSUM_ERROR_DISK`	Checksum verification error while reading from disk.	2
182	`CHECKSUM_ERROR_SD`	Checksum verification error while reading from a Source/Destination.	2

The following table describes the fields for the associated events (X indicates support):

	`CHECKSUM_ERROR_TAPE`	**`CHECKSUM_ERROR_DISK`**	**`CHECKSUM_ERROR_SD`**
Event Type	X	X	X
Tape Type	X		
Tape Barcode	X		
Drive Type	X		
Drive Name	X		
Disk Name		X	
Drive Serial Number	X		
Library Serial Number	X		
Source/Destination Name			X

	CHECKSUM_ERROR_TAPE	**CHECKSUM_ERROR_DISK**	**CHECKSUM_ERROR_SD**
Actor Name	X	X	X
Object Name[1]	X	X	X
Object Category[1]	X	X	X
Object Instances[1]	Y[2]	Y[2]	Y[2]
Media Name	X	X	
Request ID	X	X	X
Event End Time	X	X	X

[1] Object information is not provided for Repack requests.
[2] The Instance ID (number) is not known for all conditions. The ID is assigned, for a new instance, only after the final write to the destination media.

Export and Import Requests

Checksum is supported in the system export functions. In the XML file created, the checksum **Value** (`csValue`), **Source** (`csSource`) and **Type** (`csType`) are included.

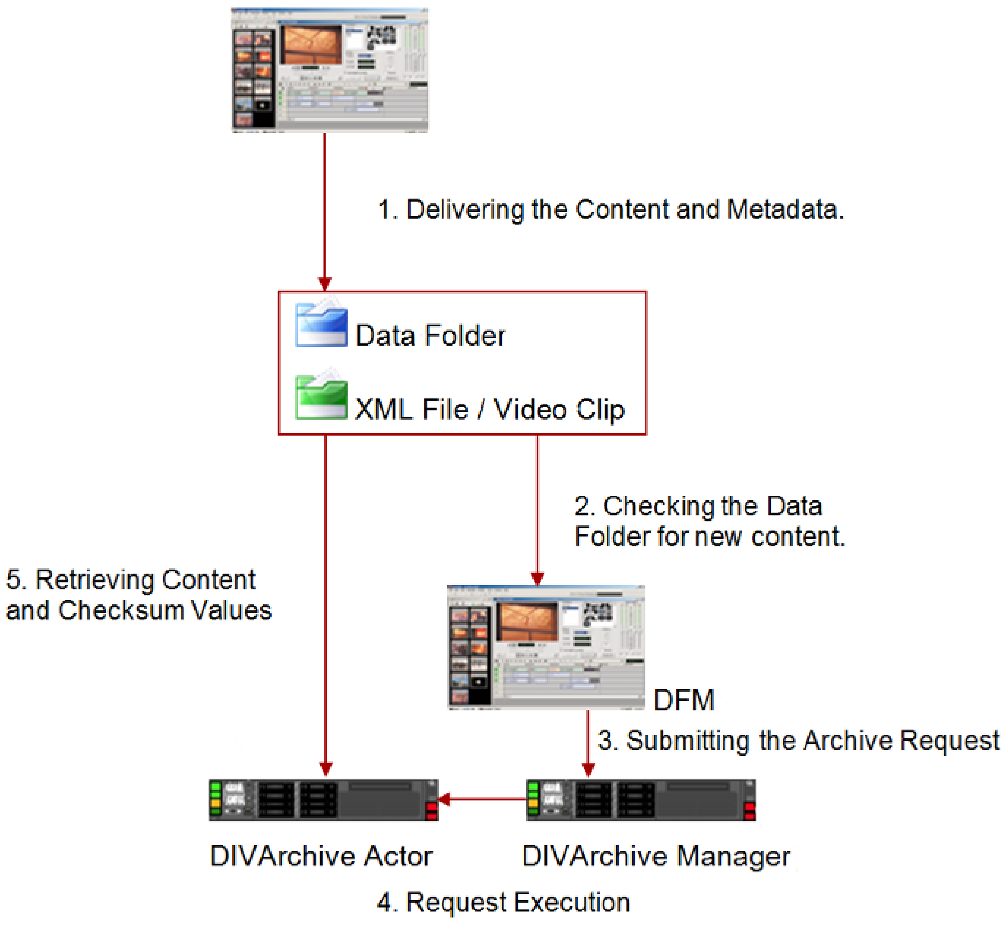
Note: The checksums are *not* verified during importing of an exported XML file. A subsequent Read operation must be performed to achieve checksum verification.

Oracle SAMMAsolo and DIVArchive Integration

Checksum includes support for SAMMAsolo integration with DIVArchive as follows:

1. SAMMAsolo will drop one video clip and one XML file into the folder monitored by the Drop Folder Monitor (DFM).
2. DFM picks up the files and analyzes them.
3. DFM submits an Archive request to the Manager.
4. The Manager sends the file location information to the Actor.
5. The Actor reads the files, reads and calculates the associated checksum, and then deletes the video clip.

   SAMMAsolo uses SHA1 as a default checksum rather than the normal MD5 default checksum. However, checksums for both algorithms are generated now so that the MD5 default is available for use in DIVArchive.


1. Delivering the Content and Metadata.
Data Folder
XML File / Video Clip
2. Checking the Data Folder for new content.
5. Retrieving Content and Checksum Values
DFM
3. Submitting the Archive Request
DIVArchive Actor
DIVArchive Manager
4. Request Execution

Genuine Checksum using a Checksum File

The **TEXT** Genuine Checksum mode allows DIVArchive to archive all files and subfolders in a specified folder while comparing their checksum values against known values stored in an external checksum file.

User Guidelines

Files that do not have a matching checksum in the external checksum file will be archived with DIVArchive's calculated checksum and the external checksum file will not be archived.

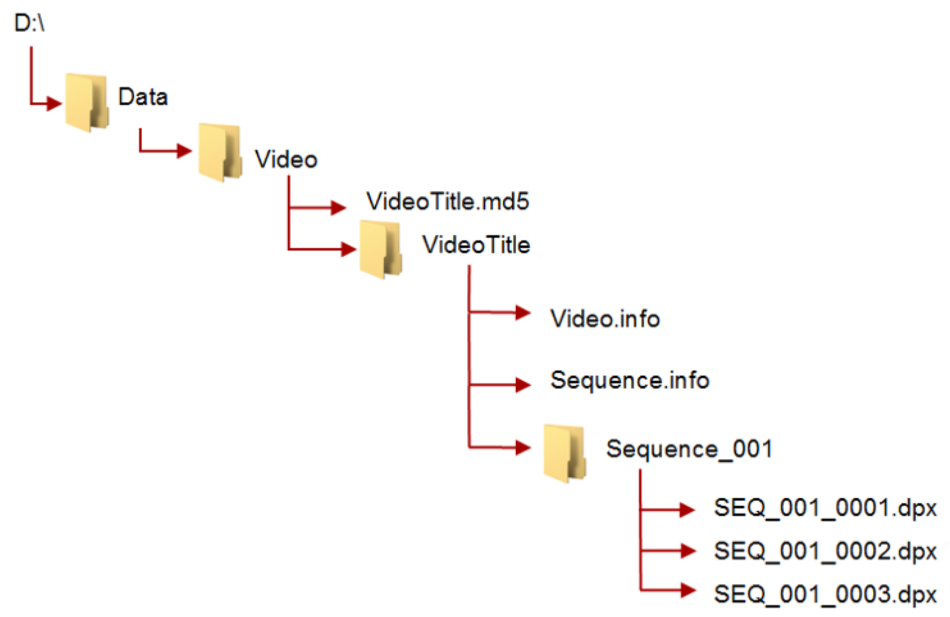
This implementation is customer-specific, supports only MD5, and checksums must be in a `.md5` text file. *Unicode is not supported.*

External Checksum File Format

The external checksum file for the **TEXT** Genuine Checksum mode follows the file format of standardized MD5 Hash File Generators (for example, md5sum). The format of each line in the external file is as follows:

```
<32-character MD5 hash><2-character whitespace>[relative path/file name]
```

The *relative path/file name* must be a path relative to the folder that is being archived. Consider the following sample file tree:



In the previous figure the external checksum file is `D:\Data\Video\VideoTitle.md5`. The contents to be archived will be every file and folder within `D:\Data\Video\VideoTitle`. The external checksum file will contain the MD5 hash and the path (relative to `D:\Data\Video\VideoTitle`) for each file formatted similar to the following:

```
9400bbcc4b97a40e2679ff1cc6941052     Video.info
237a702a03927458e1a6fc981466adf6     Sequence.info
ff58404ce3a01ca44273c8a619d4284d     Sequence_001/SEQ_001_0001.dpx
a1f1ca44204ca619d4a41273c80bbc47     Sequence_001/SEQ_001_0002.dpx
```

Requirements

- A checksum file must be present in the folder specified by the *Root File Path*.
- Checksum files must end with a `.md5` file extension.
- The folder name is associated with the checksum file's file name (excluding the md5 extension) which contains all the files that will be archived (this folder must exist).

  For example, if the checksum file is `D:\Data\Video\VideoTitle.md5` then the corresponding folder containing the files to be archived is `D:\Data\Video\VideoTitle`.
- The checksum file must be present in the folder that is the parent to the folder specified by the *Root File Path*.
- For a file to be archived with the Genuine Checksum value, the file must be referenced with a corresponding checksum within the checksum file.
- Files can be archived without a corresponding checksum within the checksum file. In this case, no checksum comparison will be performed.

DIVArchive Configuration Utility Configuration

Use the following procedure to configure Genuine Checksum **TEXT** mode using the Configuration Utility:

1. Create a new Source/Destination entry with the ***Source Type*** set to either **DISK** or **FTP_STANDARD**.
2. Specify an appropriate ***Root Path*** - this path, along with the input files, specified during the Archive request will be used to determine the location of the checksum file (see the section Selecting the Root File Path for further details).

   Examples:

   - If the ***Source Type*** is **DISK**, the ***Root Path*** can be set to `D:\Data`.
   - If the ***Source Type*** is **FTP_STANDARD**, the ***Root Path*** can be set to `/Data`.
3. Perform the following actions in the DIVArchive Configuration Utility:
   1. Set the ***External Checksum Source*** to **YES**.
   2. Set the ***Checksum Type*** to **MD5**.
   3. Set the ***GC Mode*** to **TEXT**.
   4. Click the **OK** button.
   5. Click the **Tools** menu, then click the **Notify Manager** menu item to notify the Manager of the new (or changed) configuration.

Selecting the Root File Path

The ***Root File Path*** must point to the folder containing the files to be archived. Therefore, the correct folder and file paths in the ***Source/Destination*** and Archive Request form must be set so that the checksum file can be located one folder level above. For example, if the checksum file is located in `D:\Data\Video\NewTitle.md5` (or `/Data/Video/NewTitle.md5` for **FTP_STANDARD** type), the appropriate file and folder paths can be set as follows:

This table describes Root File Paths for disks.

Source/Destination (Root Path)	Archive Request (File Path Root)	Archive Requests (Files)
`D:\`	`Data\Video\NewTitle`	`*`
`D:\Data`	`Video\NewTitle`	`*`
`D:\`	`Data\`	`Video\NewTitle\`
`D:\`		`Data\Video\NewTitle\`

This table describes Root File Paths for FTP.

Source/Destination (Root Path)	Archive Request (File Path Root)	Archive Request (Files)
`/`	`Data/Video/NewTitle`	`*`
`/Data`	`Video/NewTitle`	`*`
`/`	`Data/`	`Video/NewTitle/`
`/`		`Data/Video/NewTitle/`

Limitations

- Long path names are supported on both Windows and Linux.

- Absolute path names are supported on both Windows and Linux to a maximum of 4000 characters.
- Relative path names are limited to 256 characters on Windows systems (only).
- Only ASCII, non-UTF-8 encoded checksum files are supported.
- The format of the checksum file is that each line begins with an MD5 checksum, followed by 2 spaces, and then the file path to the referenced file. *No other formats are recognized.*

Archive Instructions

Use the following procedure to perform an Archive operation:

1. In the DIVArchive Control GUI, navigate to the Manager area.
2. Click **Action**, and then click **Archive**.
3. In the ***Source*** list, select the ***Source/Destination*** entry that was created earlier during the configuration stage.
4. Set the desired ***File Path Root*** (see Selecting the Root File Path for details).
5. In the ***Files*** field, enter in the path to the location of the files to be archived (see Selecting the Root File Path for details) and append with a wildcard symbol (an asterisk - *). Remember that the location of the checksum file is one folder above the files to be archived.
6. Enter `-r` in the ***Options*** field.
7. Fill in the rest of the entry fields in the request form and click **Send**.

DIVArchive API Support for Checksum Verification

Parameters for the `getFilesAndFolders` API call support checksum retrieval. In the following examples, only the updates associated with checksum retrieval are described. Refer to the *Oracle DIVArchive C++ API Reference Manual* in the DIVArchive Additional Features Documentation library for detailed information.

Functionality includes the following:

- Folders do not contain a checksum.
- Several checksums per file are provided (if available) including MD5, SHA1, and so on.
- The Genuine Checksum will be identified.

Synopsis

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

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

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

`checksumType`
The type of checksum, such as MD5, SHA1, and so on.

`checksumValue`
The value of the checksum in hexadecimal format.

`isGenuine`
True if this checksum was provided at the time of the archive and verified as Genuine, otherwise this is false.

Oracle DIVAnet Checksum Workflows

Oracle DIVAnet releases that coincide with DIVArchive 7.3.0 and earlier releases are considered Legacy DIVAnet releases. They are also referred to as *DIVAnet 1.0* and the *Access Gateway*.

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

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

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

The DIVArchive 7.4 API has the option to retrieve checksum values from the stored objects using the `getFilesAndFolders` call.

Since DIVAnet does not store the checksum values in the DIVAnet database, it must retrieve values from the corresponding DIVArchive system containing the object. Described below are three DIVAnet scenarios suitable for different customer configurations.

1. DIVAnet will always retrieve checksum values from the local DIVArchive system (primary processor).

   Configuration

   This scenario uses the default configuration

   Expected Functionality

   Always sending the request to the primary DIVArchive system.

   Failure Scenario

If the local DIVArchive system does not contain the specified object, the process fails and an error will be returned.

2. DIVAnet will select the DIVArchive system based on the location of the object specified in the request parameters (resource-based processor).

   Configuration

   - The following line must be placed in the `router.xml` file:

     ```
     <message class=".api.GetFilesAndFoldersMessage"
     messageProcessorClass=".ResourceBasedRoutingMessagesProcessor"
     Object="objectName,objectCategory"/>
     ```
   - The following line must be added to the `uniqueness-config.xml` file in the `<siteIdProcessingConfig>` section:

     ```
     <siteIdProcessor messageClass=".api.FilesAndFoldersMessage"
     siteId="siteName"/>
     ```

   Expected Functionality

   - Checks which DIVArchive system has the object stored and sends the request to that system. If the object is present on both the primary and secondary DIVArchive sites, the request will be sent to the primary DIVArchive site.
   - The DIVArchive site selected by DIVAnet will be returned in the *siteName* field.

   Failure Scenarios

   - If the DIVArchive *siteName* is incorrect, an error will be returned.
   - If the selected DIVArchive system does not contain the specified object, an error will be returned.

Example:

If the *SelectiveRoutingMessagesProcessor* is used for a `getFilesAndFolders` request, the message will be sent to the DIVArchive system specified in the *options* parameter of the `getFilesAndFolders` request. When specifying this parameter, DIVArchive's *siteName* from the Access Gateway Configuration File (`AccessGateway.conf`) will be used.

The `AccessGateway.conf` file contains the following DIVArchive Manager configurations:

```
<managerConnections>0
  <address siteName="diva1_local" host="127.0.0.1" port="9007" />
  <address siteName="diva2_remote" host="172.16.4.218" port="9008" />
</managerConnections>
```

- To send a `getFilesAndFolders` request to a local DIVArchive system (ip: `127.0.0.1`, port: `9007`), `diva1_local` should be the value for the *options* parameter.
- To send a `getFilesAndFolders` request to a remote DIVArchive system (ip: `172.16.4.218`, port: `9008`), `diva2_remote` should be the value for the *options* parameter.

Genuine Checksum through AXF Transfer

The AXF Genuine Checksum Mode allows DIVArchive to archive all files and subfolders in a specified AXF file, while comparing their checksum values against known values stored in the AXF file. This type of workflow is typically combined with a Restore request with the **-axf** keyword in the ***Request Options***. The requirements to perform this type of checksum verification are as follows:

- The AXF containing the files to be archived must contain checksum information for each file.
- The AXF must contain checksums of the expected type (specified in the configuration).

DIVArchive Configuration Utility Settings

Use the following procedure to configure the DIVArchive settings in the Configuration Utility for using GC in AXF transfers:

1. Create a new ***Source/Destination*** entry with the ***Source Type*** set to either **DISK**, **FTP_STANDARD**, or **EXPEDAT** as appropriate.

   If required specify an appropriate ***Root Path*** - this path along with the input files specified during the Archive request, will be used in determining the location of the checksum file.

   Examples:

   - If the ***Source Type*** is **DISK**, the ***Root Path*** can be set to `D:\root`.
   - If the ***Source Type*** is **FTP_STANDARD**, the ***Root Path*** can be set to `/root`.
2. Set the ***External Checksum Source*** to **YES**.
3. Set the ***Checksum Type*** to the expected checksum type (for example MD5).
4. Set the ***GC Mode*** to **AXF**.
5. Click the **OK** button.
6. Click the **Tools** menu, and then **Notify Manager** to notify the Manager of the new (or changed) configuration.

Limitations

- This workflow only works with AXF requests generated by DIVArchive.
- Verify Following Restore (VFR) is not compatible with the **-axf** option.
  - VFR was designed to read back the restored content from a video server to verify it has not been corrupted. Using the **-axf** option does not create a *real* restore, rather an *object export* in an AXF wrapper. These options are mutually exclusive and must not be part of the same workflow.

Archive Instructions

Use the following procedure to perform an Archive operation:

1. In the DIVArchive Manager, click **Action**, and then **Archive**.
2. In the ***Source*** list, select the **Source/Destination** entry that was created in the configuration stage.
3. Set the desired ***File Path Root***.
4. Enter the path to the location of the AXF file in the ***Files*** field. The extension of this file must be `.axf`.
5. Fill in the remaining entries in the request form and then click **Send**.

Genuine Checksum through an Archive Request

The Genuine Checksum (GC) may be passed as part of the Archive Request API command. This may be used with a C++, Java, or WS API. To use this feature, include the ***GC Checksum Value*** for each file in the request and set the option in the ***Options*** field. The requirements to perform this type of checksum verification are as follows:

- The `-gcinfilelist` option must be specified either in the Archive request or the source (in the Manager configuration). This parameter must not be used with Source/Destinations that have Genuine Checksum enabled (read from a file).
- The ***GC Type*** must match the configured default checksum type specified in the DIVArchive configuration.

File Names List Formatting

The GC Value is included as part of the file list within the Archive request using the *fileNamesList* parameter. A colon separates the file name and checksum fields as shown in the following example:

Note: The example presented uses the WS API. Other APIs will have different parameter formatting (not XML).

```
<archiveObject>
  <sessionCode>test</sessionCode>
  <objectName>OBJ1</objectName>
  <objectCategory>CAT</objectCategory>
  <source>localftp</source>
  <mediaName>array1</mediaName>
  <filesPathRoot>movies/subdir</filesPathRoot>
  <fileNamesList>test1.txt:a6f62b73f5a9bf380d32f062f2d71cbc</fileNamesList>
  <fileNamesList>test2.txt:96bf41e4600666ff69fc908575c0319c</fileNamesList>
  <priorityLevel>50</priorityLevel>
  <comments>test</comments>
  <archiveOptions>-gcinfilelist MD5</archiveOptions>
</archiveObject>
```

Archive Request Option

You use the **`-gcinfilelist`** `[type]` parameter in the Archive request. The `type` is the same as the type specified by the ***Manager: Default Checksum Type*** in the DIVArchive configuration (this is MD5 by default).

The **-gcinfilelist** keyword 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.

Limitations

- The checksum data must be formatted as `filename:123456789abcdef`. No spaces are allowed after the file name.
- Wildcards for file names are not allowed. If a wildcard is provided in a file name and a Genuine Checksum is passed, the request will be rejected.
- A maximum of one thousand files can be archived with this feature.

- This feature only works with non-complex object archives.
- Empty checksums (a colon with no values or just a space) are not allowed and result in an Invalid Checksum error.
- If a GC error occurs during the archive, error text in the `getRequestInfo/getFinishedList` call indicates that a GC error has occurred.
- If the archive is transmitted on a Source/Destination where the GC has been configured, the GC functionality of the Source/Destination will override the GC in the request.
- This option may be specified in the ***Source/Destination*** options of a Source/Destination (similar to most other options in the DIVArchive Manager).

4 Frequently Asked Questions

This chapter includes some common questions and answers, and recommendations.

How do I access the system in Engineering Mode?
Contact Oracle Support to access the system in Engineering Mode. Engineering Mode is accessible for Oracle Support personnel only to avoid accidental misconfiguration of the system, which could possibly result in degradation of DIVArchive operations and (or) loss of data.

How will different Checksum Settings affect the system performance?
The different settings affect the performance of the system as follows:

Verify Read (Default)
Requires an additional 0% to 3% processing time for the request.

Verify Write
Requires an additional 3% to 50% processing time for the request.

Verify After Archive
Requires an additional 50% to 100% processing time for the request.

Verify After Restore
Requires an additional 50% to 100% processing time for the request.

How is Checksum Support enabled and disabled?
See Chapter 2 for information on enabling and disabling DIVArchive Checksum Support.

How is Checksum Support enabled after an upgrade?
Checksum Support is enabled by default with the Verify Read (VR) workflow.

A

Tape Repack and Verify Limitations

The repack request will fail if the tape contains any corrupted objects and (or) the object fails checksum verification. You must manually resolve the conflict before performing the repack.

A checksum will not be generated for any spanned objects during Repack or Verify Tape requests. The Actor will identify any spanned files and the Manager will not attempt to verify them. A warning event will be displayed stating that a checksum was not generated or verified for the spanned content. In this case, the repack operation will not be aborted. However, the object instance will be marked as **Not Verified**.

Glossary

Drop Folder Monitor (DFM)

The Drop Folder Monitor monitors Drop Folders in the system. When new files are detected, DFM sends an operational request to the Oracle DIVArchive Manager.

Checksum Support and Checksum Types

A mathematical value computed from a group of data being transmitted and transferred with the data. The receiving device compares the checksum with its own computation. If it differs from the received checksum, it either requests the transmitting device to resend the data, or generates an error. Each checksum has a specific algorithm having its own level of verification.

Oracle SAMMAsolo

A single-stream migration solution that supports a wide variety of video tape formats on input and simultaneously delivers multiple digital files (in real time) including frame accurate loss-less preservation quality, broadcast quality mezzanine, and proxy files. Numerous formats are available and SAMMA can produce up to five different simultaneous digital output files in real time.