Webservices Documentation

Miria 4.0

Publication Number: Miria-ADMWS-PDF-EN-0323-REV1

Publication Date: March 2023

Atempo SAS
23 Avenue Carnot
91300 Massy - France

Contents

CHAPTER 1 - Integration Using Web Services

This topic describes the Web services provided by Atempo that you can use to perform some Miria operations in your production environment through HTTP requests.

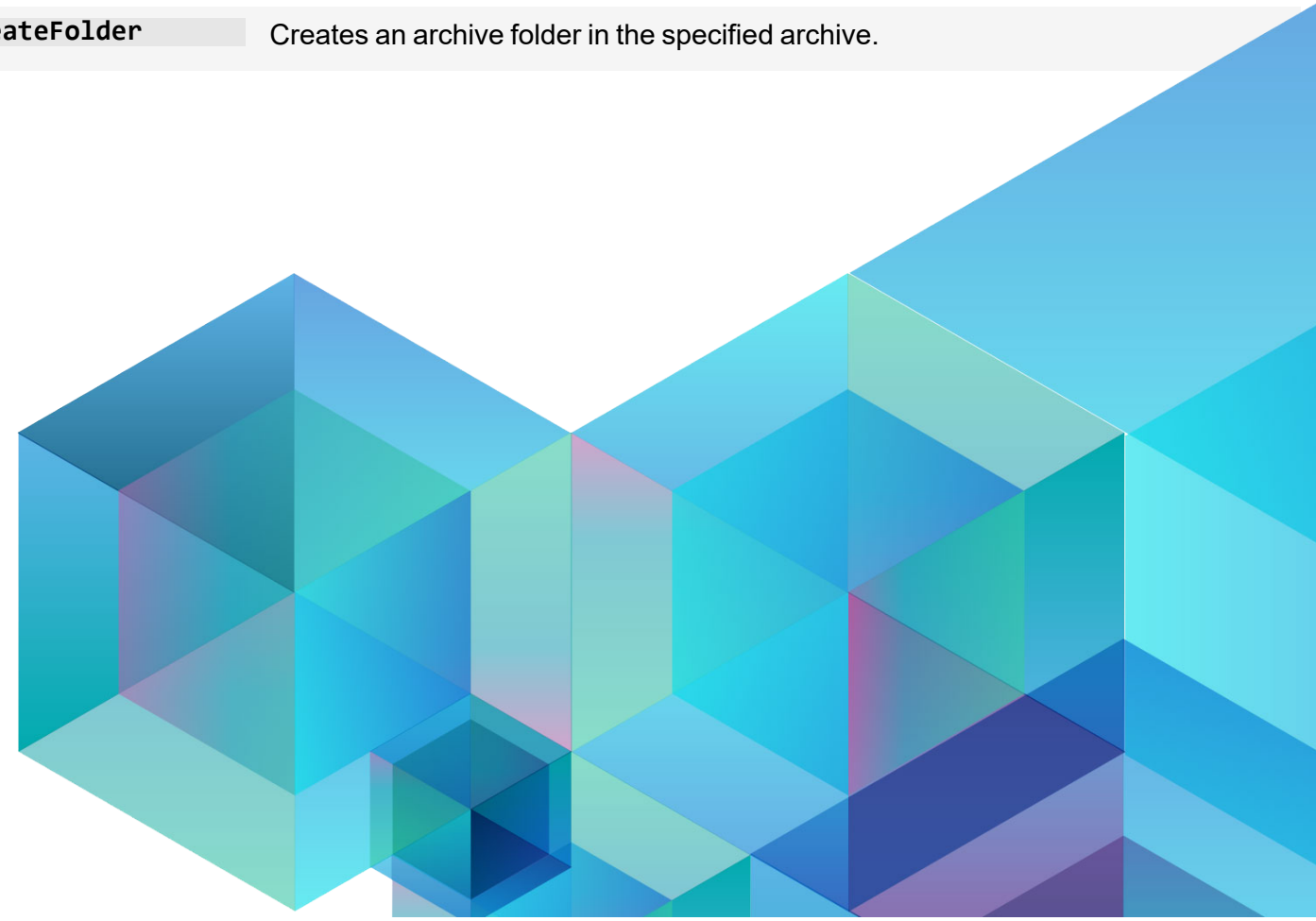
These topics are intended for developers who are familiar with developing Web services.

Miria Web Services

Web services enable third-party applications to make HTTP requests to Miria to perform functions such as archiving data, getting status, or retrieving data. Onced the task performed, Miria sends its response back to the requesting application as an XML file.

This table describes Miria Web services that you can use with HTTP requests:

Web Service	Description
`archiveFile`	Archives a file to the specified archive folder.
`cp`	Copies files and directories from a platform to another.
`fileSearch`	Search for files in the archive project based on criteria provided
`deleteFile`	Marks as deleted, file(s) from an archive project.
`fileCheck`	Checks if specified file(s) exist(s) in the archive project.
`createFolder`	Creates an archive folder in the specified archive.



Web Service	Description
`archiveFolder`	Archives recursively a folder and its sub-objects to the specified archive folder.
`deleteObject`	Deletes an archived file or directory.
`getDeviceList`	Obtains the list of all libraries and drives connected to a given platform.
`getDriveStatus`	Obtains the status of the drive managed by the specified Media Manager storage manager.
`getFileStatus`	Obtains the status of the specified archived file.
`getJobStatus`	Obtains the status of the specified job.
`getLibraryStatus`	retrieves the status of the Media Manager libraries.
`getPlatformStatus`	retrieves the status, type, and operating system of a Miria platform.
`getMediaList`	Obtains the list of media based on some criteria.
`getServerInfo`	Returns information about the Miria server installation (server name, HTTP port, HTTPS port).
`ls`	Lists the contents of an archive folder. Obtains the name and type of each archived object.
`reportVolume`	Generates a report of the volume of archived data.
`retrieveFile`	Retrieves an archived file to the specified destination.

Web Service	Description
`retrieveFolder`	Enables to restore the entire tree structure of a Folder or Directory present in an archive projet.
`dbMonitor`	Displays the information o the database monitoring.
`launchTask`	Launches the specified ta .
`getJobEvents`	Displays t e en f job.
`duplicateArchive`	Duplicates an existing project archive to a new project archive set. Only its settings are duplicated, not its objects.
`verifyMedia`	Starts a verification job on a media.

Implementing Web Services

To implement Web services in your company, you must perform these two steps:

Generating the URL

To make Miria requests through the Web services, you must build the basic `ADA:WS` Web service URL, to which you then add the different commands and options.

This URL is built using the `ada_service` command and does not require any preliminary configuration. The URL is based on the database and user name information, and by default is encoded using the Miria version. Hence, the URL expires at each Miria upgrade, and you must generate a new one.

To keep the same URL regardless of Miria version, set the `url_passcode` tunable before generating it. The URL is encoded using a passcode rather than the version.

See Tunables and Environment Variables for details on the `url_passcode` tunable.

To build the basic URL

1. On the Miria server or agent:
   Windows. Select Start › All Programs › Miria › Miria Environment Command Prompt.
   macOS. - Unix. Open a terminal in the `$ADA_HOME$/Binary` directory and set the Miria environment by running the `. .ADA.sh` command.
2. Type the command:

- **HTTP.** `ada_service -build_[illegible]:password`
- **HTTPS.** `-build_url_s -id[illegible]`

Use `-build_url_s` to generate a secure URL.

> **Note**: To use the `-build_url_s` option, you must have configured the Miria server to accept HTTPS connections. See the Installation Documentation for details.

The command returns the URL corresponding to your environment, as well as this additional information:

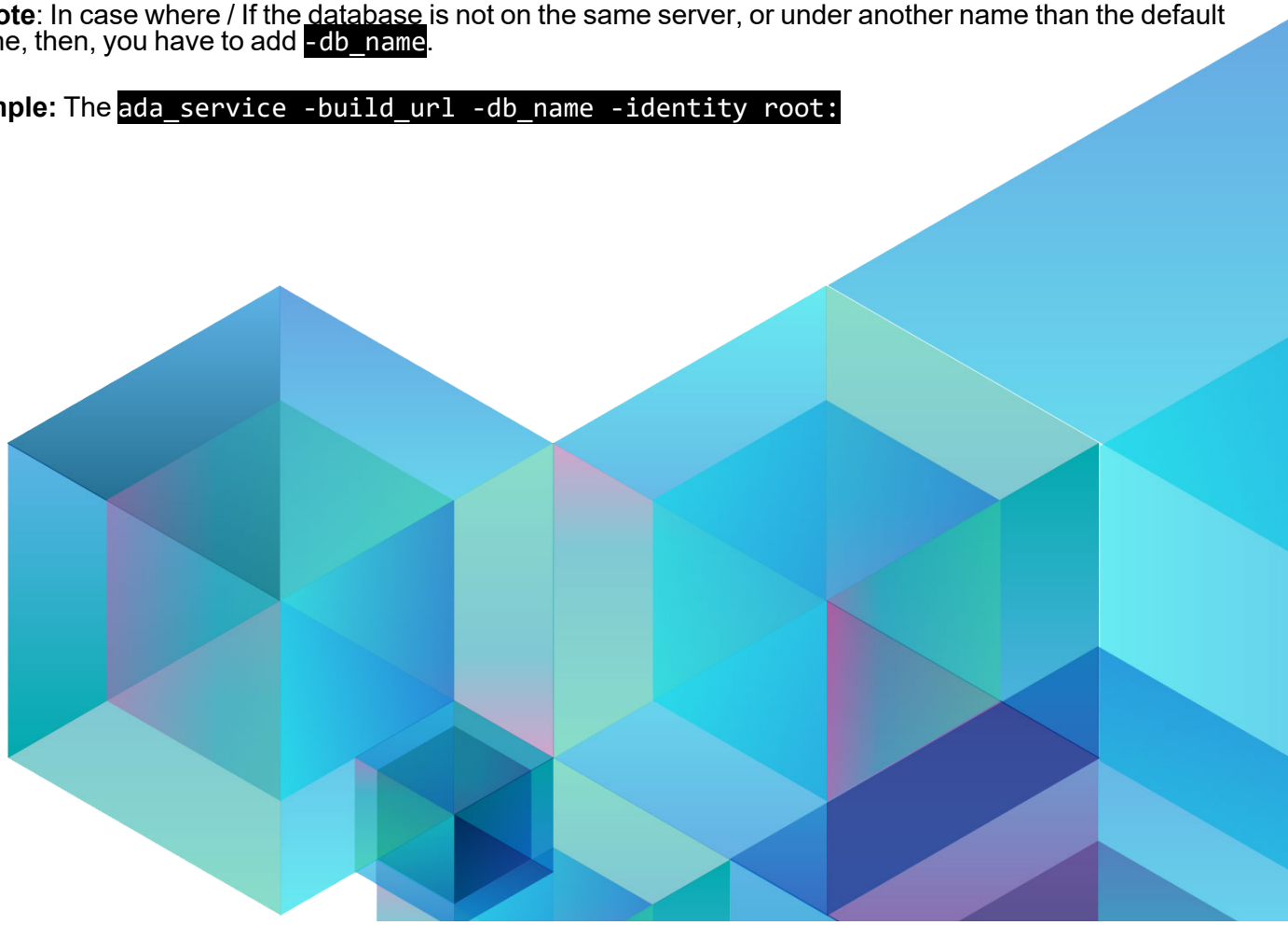
- The user who built the URL
- The passcode used to encode the URL
- The protocol supported by the URL

Example. The `ada_service -build_url -identity root:` command returns this information:

```
https://miria-server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b766531/ADA/WS
Available methods in ADA:WS package are : deleteFile, fileSearch, getServerInfo,
getJobStatus, retrieveFile, retrieveFolder, archiveFile, archiveFolder, deleteObject,
getFileStatus, getDriveStatus, getLibraryStatus, getPlatformStatus, ls, createFolder,
getMediaList, cp, reportVolume, getDeviceList, dbMonitor, launchTask, fileCheck
User: root
Passcode used: 3_10_1_7
Protocol Supported: HTTP
```

> **Note**: In case where / If the database is not on the same server, or under another name than the default one, then, you have to add `-db_name`.

Example: The `ada_service -build_url -db_name -identity root:`



Running the Web Service Requests

The Miria server handles all the Web service HTTP requests. You can run these requests from a:

- **Web browser.** Enter the request in the address bar of a Web browser anywhere on the network.

Or

- **Third-party application set to run HTTP requests.** Configure your application to run the Miria requests.

HTTP Request

To make an HTTP request, use this syntax:

```
<URL><operation>?<option1>=<value1>&<option2>=<value2>...
```

This table describes the parameters that you must configure to make an HTTP request:

Parameter	Description
`<URL>`	URL corresponding to your Miria environment. First generate this URL as described in Generating the URL.
`<operation>`	Name of the Web service operation that you are requesting, such as `archiveFile`.
`<option1>`	Name of the operation option, such as `src` to indicate the source file.
`<value1>`	Value of the option, such as `C:\Temp\rules.xml` to indicate the file path. The value must be encoded according to the RFC 3986 (e.g., a space must be replaced by `%20`.)

Important: An operation succeeds only if the user specified when building the URL is granted the corresponding right in Miria (e.g., to run the `archiveFile` request, the user must be granted the *Archive* right).
See Creating a Project Archive and List of Default and Advanced Settings for details.

XML Result

This HTTP request returns as a result an XML document displayed in the Web browser. You can either read it in the Web browser or configure an application to interpret the XML result.

Example. To get the status of job 151, run this request:

```
https://miria-
server/meta/BD117E7D5CF0912AE[illegible]b736531/ADA/WS/getJobStatus?job_id=151
```

The results are displayed in an XML document (Figure 1).

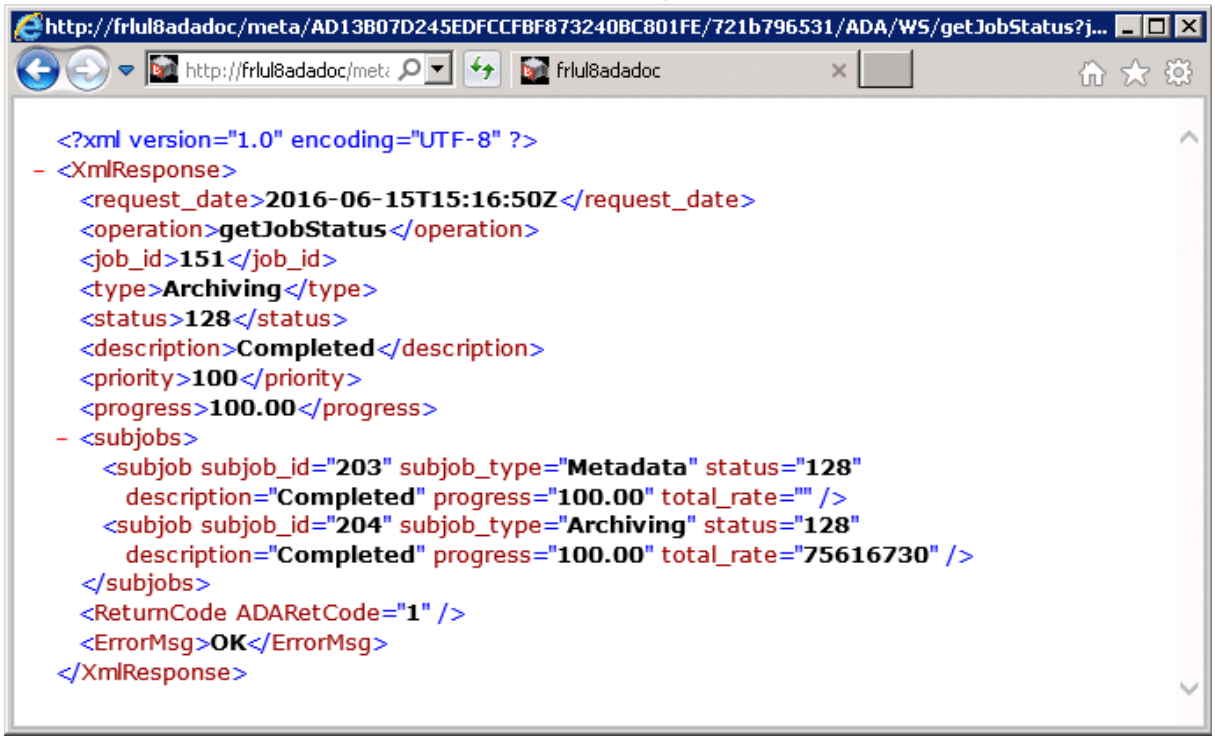


Figure 1: Results of the HTTP request

CHAPTER 2 - Archiving a File

To archive a file, apply this procedure.

Operation

`archiveFile` archives a file in a Miria archive folder.

Options

These are the options that you can use with the `archiveFile` command:

- `src=[host@]path_on_OS` indicates the path of the file that you want to archive. The host specified by `[host@]` must be a Miria agent.

  You do not have to provide the host name if the file to archive is located on the Miria server.
- `dst=archive@path_in_archive` indicates the path of the archive folder in which you want to archive the file. If the archive folder does not exist, Miria creates it automatically.
- `debug=1` enables debug traces in the Miria Events.

Advanced Settings Options

These options enable you to apply some advanced settings to the archiving job. They override the settings defined in the Administration Console.

See Archiving Settingsfor details on settings related to archiving.

These are the Advanced Settings options that you can apply:

- `archiving_policy=`*`policy_name`* specifies that the same archiving policy applies to both Windo and Unix or macOS.
- `windows_archiving_policy=`*`policy_name`* specifies the Windows archiving policy t

- `unix_archiving_policy=policy_name` specifies the Unix or macOS arc
- `custom_media_rule=rule_name` specifies the custom media rule to be used.
- `parallel_class_archiving=1` creates multiple streams for simultaneous archiving of several files or directories.
- `collect_metadata=1` collects audiovisual and image metadata during archiving so that you can use it for searches on archives.
- `collect_pr_metadata=1` collects partial retrieval metadata during archiving.
- `collect_mime=1` collects the MIME type of files during archiving.
- `prevent_dir_spanning=level` indicates the directory level that you want to archive as a whole on a single media.

  Use this setting when archiving with Media Manager to avoid file tree splitting and make retrieval from the media file system easier. The value is an integer between `0` and `99`.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/archiveFile?src=macdc@C:
\Temp\rules.xml&dst=Documentation@/Atempo%20Media%20Manager/V32SP3
```

XML Result

The `archiveFile` result is returned once the archiving job is initiated, without waiting for the job to be completed. To know the archiving job status, use the `getJobStatus` Web service.
See Obtaining a Job Status for details.

```
<?xml version="1.0" encoding="UTF-8"?>
<XmlResponse>
<request_date>2011-09-13T15:53:13Z</request_date>
<operation>archiveFile</operation>
<job_id>863</job_id>
<ReturnCode ADARetCode="1"/>
```

<ErrorMsg/>
</XmlResponse>

CHAPTER 3 - Searching for files

To search a file, apply this procedure.

Operation

`fileSearch` searches for files in the archive project based on criteria provided.

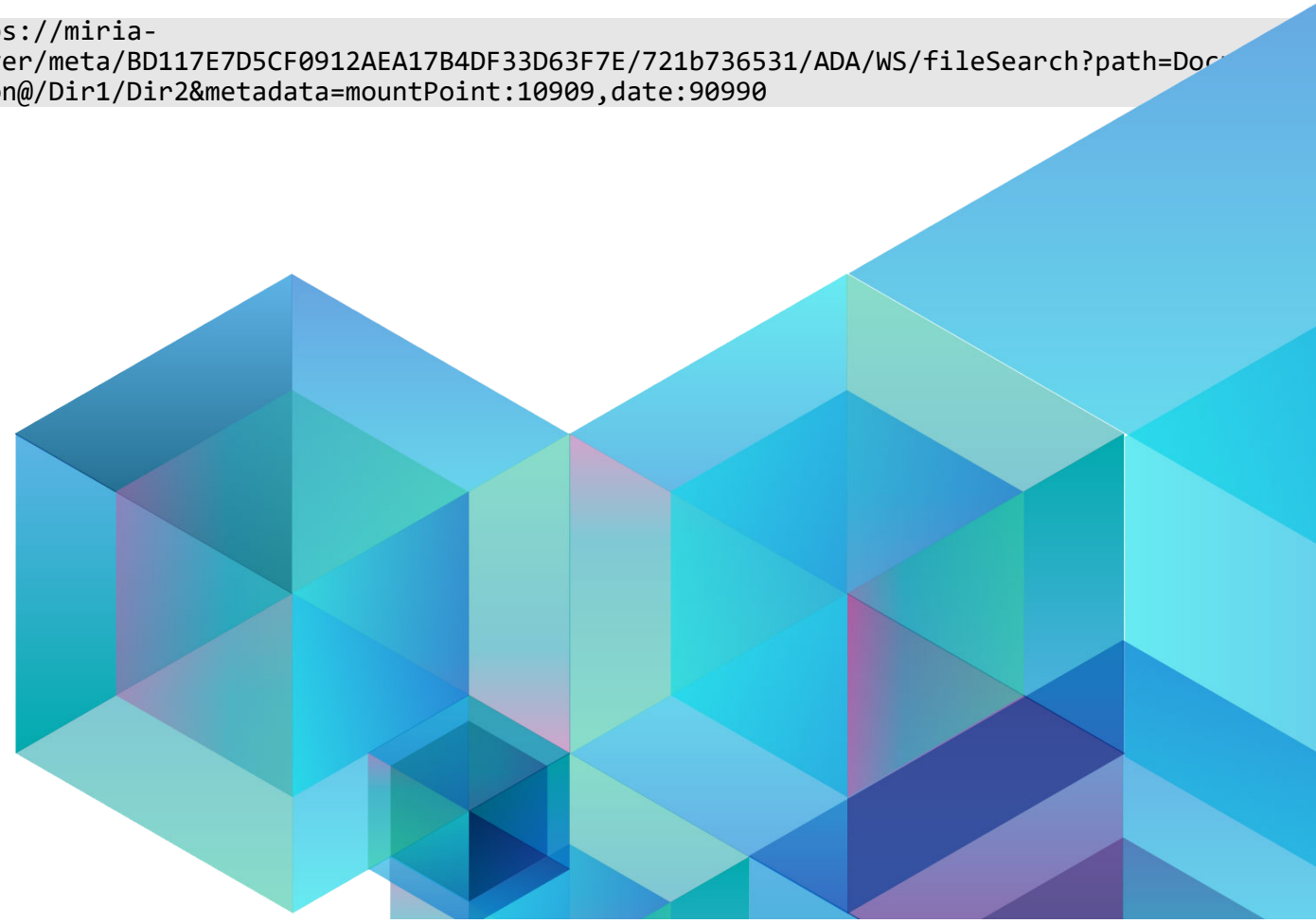
Options

These are the options that you can use with the `fileSearch` command:

- `metadata=key:value,key:value,` … describes specific metadata the user wants to use to search files. This option could be combined with others.
- `filename=file.docx` is the name of the file in the archive the user is searching for. % could be used as wildcard character. This option could be combined with others.
- `path=ArchiveName@/dir1/dir2/` indicates on which path we restrict the operation.
- `start_date=yyyymmdd` indicates the archiving starting date. This option can be used alone or combined with the end_date. This option lets the user specify a date range to search for files. This option could also be combined with others.
- `end_date=yyyymmdd` indicates the archiving end date. This option can be used alone or combined with the start_date. This option could also be combined with others.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/fileSearch?path=Doc
ation@/Dir1/Dir2&metadata=mountPoint:10909,date:90990
```



XML Result

```
<?xml version="1.0" encoding="UTF-8"?>
<XmlResponse>
<request_date>2011-09-13T15:53:13Z</request_date>
<operation>fileSearch</operation>
<ReturnCode ADARetCode="1"/>
<Results number=”5”/>
<filesList>
<file Name=”archive@Documentation:/Dir1/Dir2/file.docx”/>
<file Name=”archive@Documentation/Dir1/Dir2/file2.docx”/>
<file Name=” archive@Documentation/Dir1/Dir2/file3.docx”/>
<file Name=” archive@Documentation/Dir1/Dir2/file4.docx”/>
<file Name=” archive@Documentation/Dir1/Dir2/file5.docx”/>
</filesList>
<ErrorMsg/>
</XmlResponse>
```

CHAPTER 4 - Deleting a file

To delete a file, apply this procedure.

Operation

`deleteFile` marks file(s) from an archive project as deleted.

Options

These are the options that you can use with the `deleteFile` command:

- `file=ArchiveName@filePathInArchive` indicates the path of the file in the archive folder to mark as deleted.
- `filesList=[host@]path_on_os` indicates the path of a file containing the list of files to mark as deleted. This option replaces and overrides previous ones. To be valid, this file must contain one filePathInArchive by line. The host specified by `[host@]` must be a agent or a platform name.
- `path=ArchiveName@/dir1/dir2/` indicates on which path the operation is restricted. This option replaces and overrides `file` & `filesList` options, but should be combined with one or many of the following options: metadatas, start_date, end_date or filename.
- `metadata=key:value,key:value,` … describes specific metadata to use to identify files to mark as deleted. This option should be used with path option, but it could be combined with others.
- `start_date=yyyymmdd` indicates the archiving starting date. This option can be used alone or combined with the end_date option. This option allows to specify a date range to identify the deleted target. This option replaces and overrides `file` & `filesList` & `path options`, but it could be combined with others.
- `end_date=yyyymmdd` indicates the archiving end date. This option can be used alone or combined with the start_date option. This option replaces and overrides `file` & `filesList` & `path options`, but it could be combined with others.

- `filename=filename:` used with the option path to indicate a simple name e used as wildcard character. This option deletes the filename recursively through folder and subfolder of the archive tree, indicated by the option path.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AE 17 4D D 3F 72 b736531/ADA/WS/deleteFile?file=Document
ation@Dir1/Dir2/file.docx
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2011-09-13T15:53:13Z</request_date>
<operation>deleteFile</operation>
<ReturnCode ADARetCode="1" />
<ErrorMsg />
</XmlResponse>
```

CHAPTER 5 - Checking the existence of files

To check if a file exist, apply this procedure.

Operation

`fileCheck` checks if specified file(s) exist(s) in the archive project.

Options

These are the options that you can use with the `fileCheck` command:

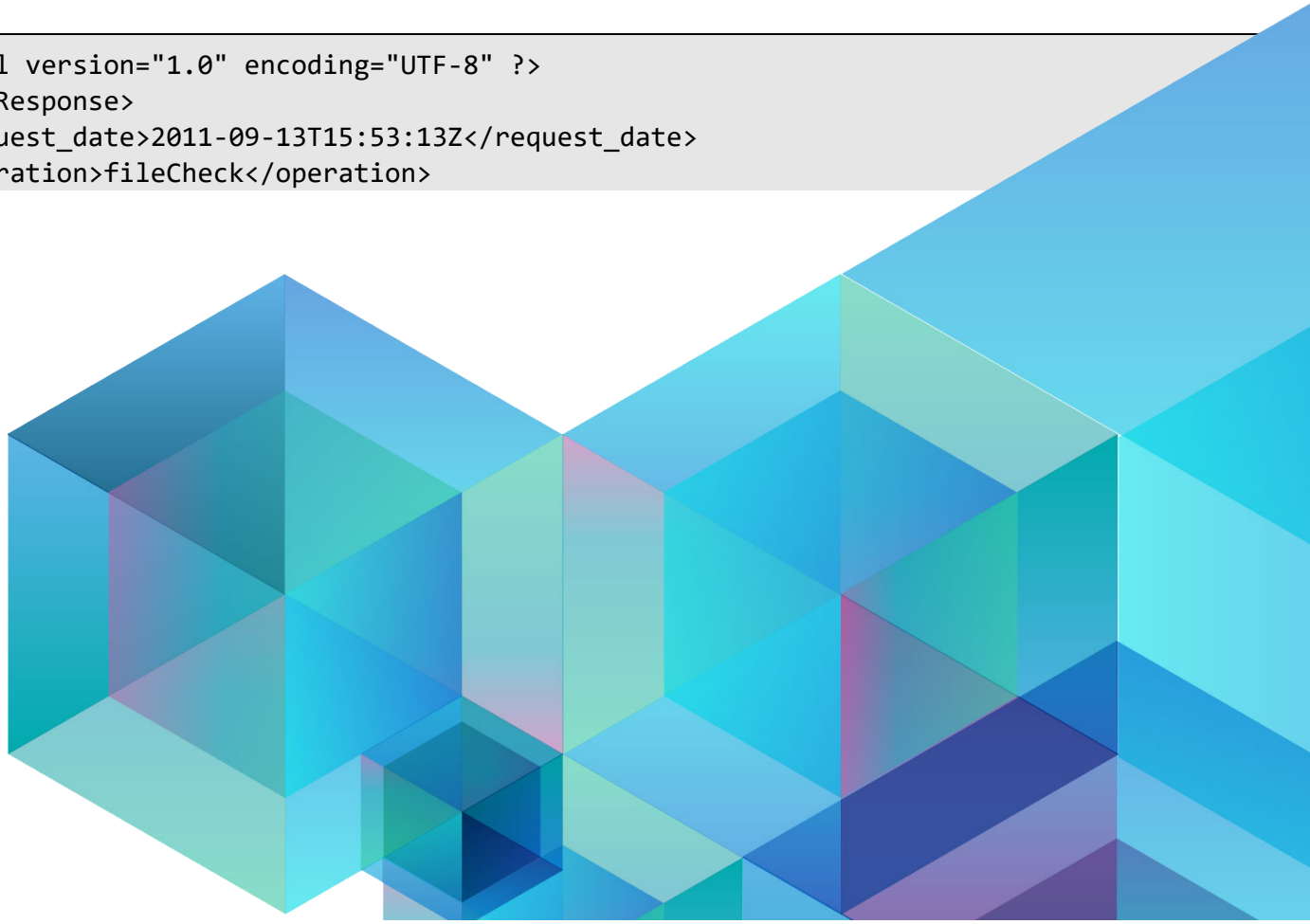
- `file=ArchiveName@filePathInArchive` indicates the path of the file in the archive to check.
- `filesList=[host@]path_on_os` indicates the path of a file containing a list of files to check. This option replaces and overrides previous ones. To be valid, this file must contain one filePathInArchive by line. The host specified by `[host@]` must be an agent or a platform name.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/fileCheck?file=Documenta
tion@Dir1/Dir2/file.docx
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2011-09-13T15:53:13Z</request_date>
<operation>fileCheck</operation>
```



```
<existingFiles>1</ existingFiles>
<notexistingFiles>0</ existingFiles>
<filesList>
<file Name=”/Dir1/Dir2/file.docx”>
<status>1</status>
</file>
</filesList>
<ReturnCode ADARetCode="1" />
<ErrorMsg />
</XmlResponse>
```

CHAPTER 6 - Creating an Archive Folder

To create an archive folder, apply this procedure.

Operation

`createFolder` creates an archive folder in a Miria archive.

Options

These are the options that you can use with the `createFolder` command:

- `src=archive@path_in_archive` indicates the path and name of the archive folder that you want to create.
- `debug=1` enables debug traces in the https://miria-server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/fileCheck?file=Documentation@Dir1/Dir2/file.docx Events.

HTTP Request

```
https://miria-server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/createFolder?src=Documentation@/Atempo%20Digital%20Archive/V32SP7
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2011-09-13T16:21:44Z</request_date>
```

```
<operation>createFolder</operation>
<ReturnCode ADARetCode="1" />
<ErrorMsg />
</XmlResponse>
```

CHAPTER 7 - Archiving a folder

To archive a folder, apply this procedure.

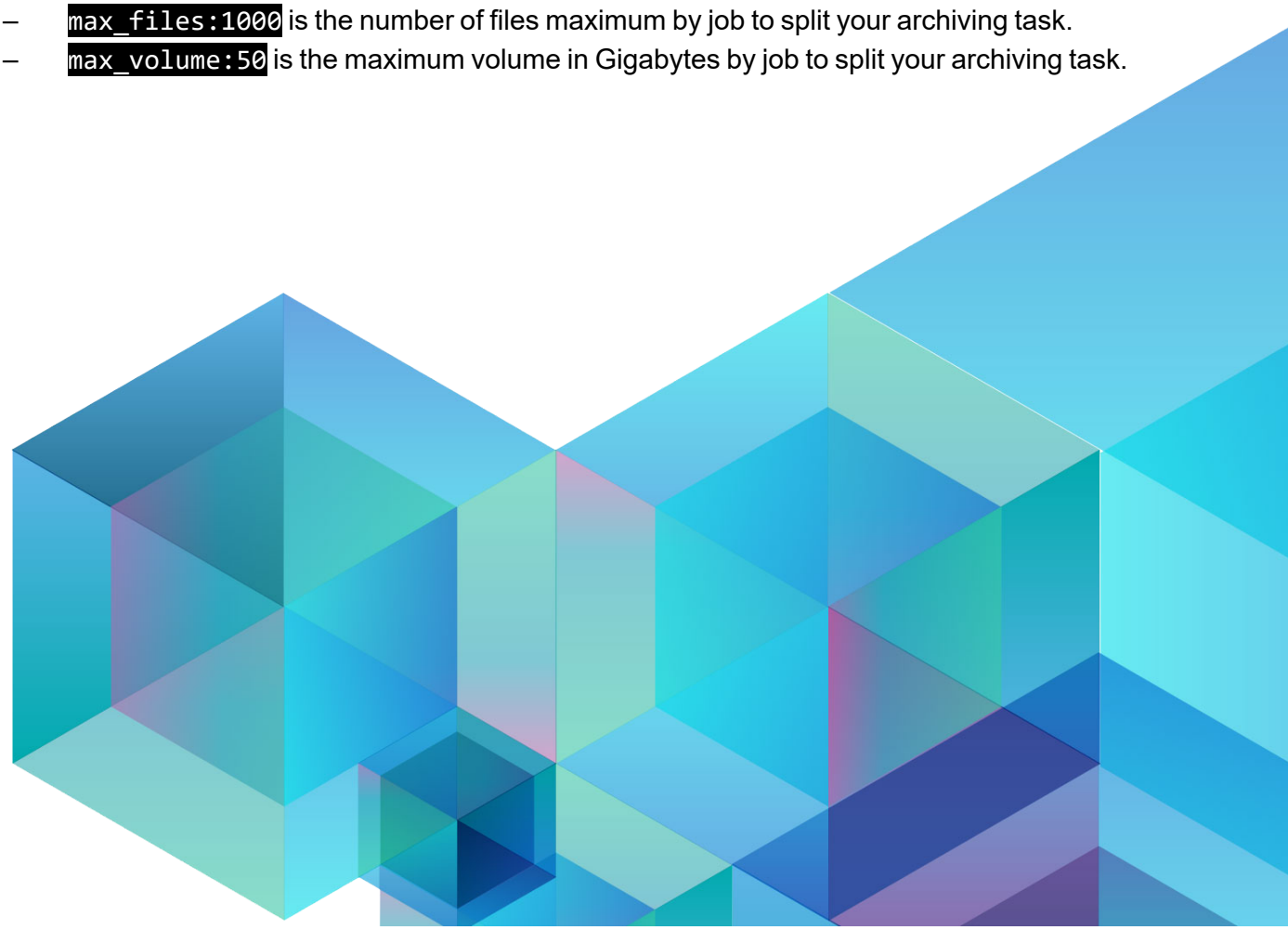
Operation

`archiveFolder` archives recursively a folder and its sub-objects to the specified archive folder.

Options

These are the options that you can use with the `archiveFolder` command:

- `src=[host@]path_on_OS` indicates the path of the folder you want to archive. The host specified by `[host@]` must be a Miria agent or a platform name.
- `dst=ArchiveName@path_in_archive` indicates the path of the archive folder in which you want to archive the file. If the archive folder does not exist, https://miria-server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/createFolder?src=Documentation@/Atempo%20Digital%20Archive/V32SP7 creates it automatically.
- `metadata=key:value,key:value,` ... describes specific metadatas the user wants to apply to archived files.
- `parallelization_rules=jobs:5,max_time:30,max_files:1000,max_volume:50` describes parallelization rules to apply for archiving tasks splitting. Jobs and media overrides all other options, but you can combine options number & volume:
  - `jobs:5` is in how many job numbers you want to split your archiving task.
  - `max_time:30` is the amount of minutes you want before to split your archiving task.
  - `max_files:1000` is the number of files maximum by job to split your archiving task.
  - `max_volume:50` is the maximum volume in Gigabytes by job to split your archiving task.



Advanced Settings Options

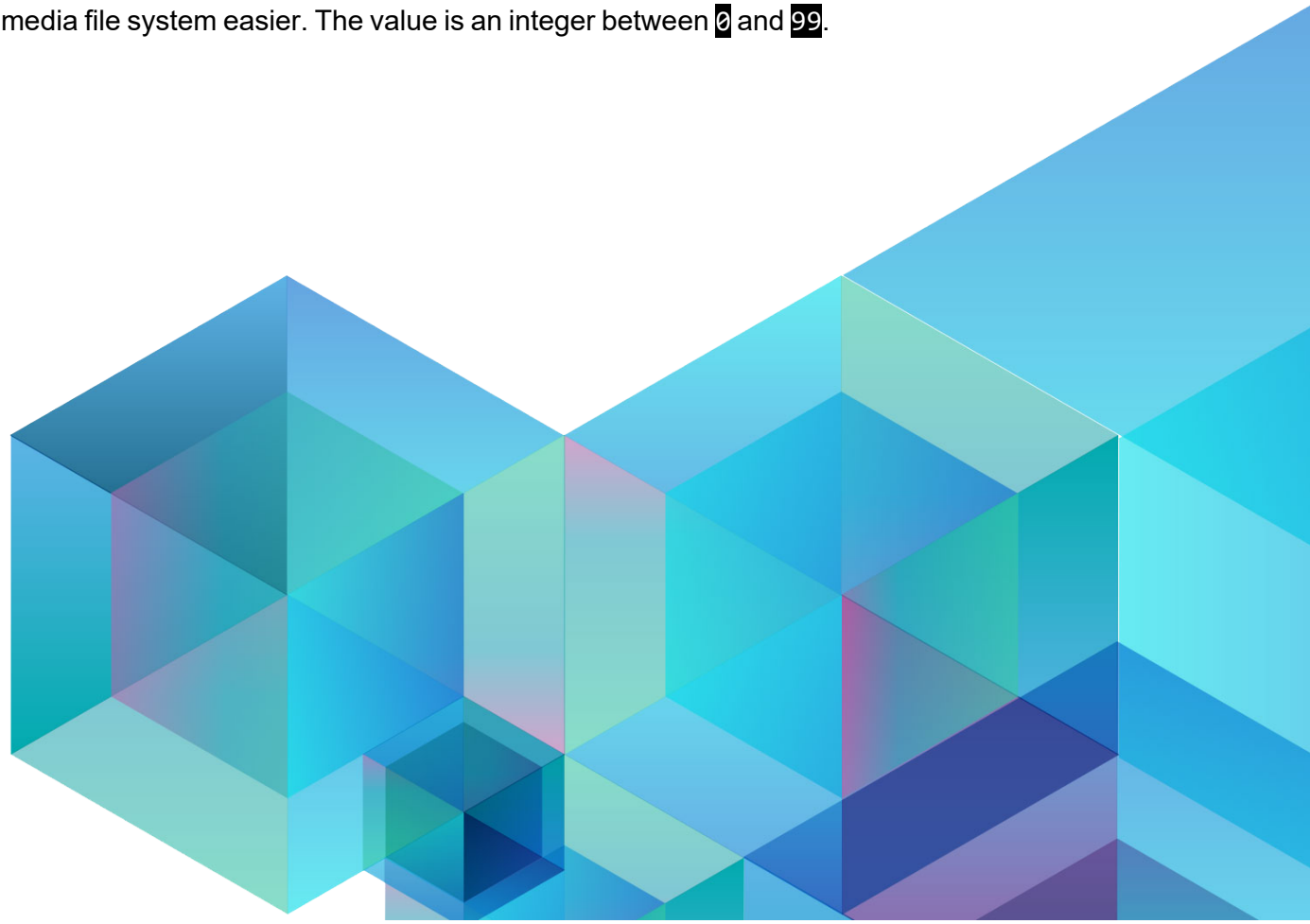
These options enable you to apply some advanced settings to the archiving job. They override the settings defined in the Administration Console.

See Archiving Settings for details on settings related to archiving.

These are the Advanced Settings options available for this type of job:

- `archiving_policy=policy_name` specifies a global archiving policy for all type of system (Windows and Unix or macOS).
- `windows_archiving_policy=policy_name` specifies the Windows archiving policy to be used.
- `unix_archiving_policy=policy_name` specifies the Unix or macOS archiving policy to be used.
- `custom_media_rule=rule_name` specifies the custom media rule to be used.
- `parallel_class_archiving=1` creates multiple streams for simultaneous archiving of several files or directories.
- `post_archiving=no_action|0| file_deletion|1| hsm_cli|2| hsm_fd|3` specifies which action to execute on the source files after archiving. The name of the action or the number can be used to execute the post action. These are the available actions:
  - `no_action|0` (default value) indicates that Miria performs no action on the source files.
  - `file_deletion|1` indicates that Miria deletes files from the source as soon as they are sent to the Miria server.
- `collect_metadata=1` collects audiovisual and image metadata during archiving so that you can use it for searches on archives.
- `collect_pr_metadata=1` collects partial retrieval metadata during archiving.
- `collect_mime=1` collects the MIME type of file format during archiving.
- `prevent_dir_spanning=level` indicates the directory level that you want to archive as a whole on a single media.

  Use this setting when archiving with Media Manager to avoid file tree splitting and make retrieval from the media file system easier. The value is an integer between `0` and `99`.



HTTP Request

```
            https://miria-server

/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721736531/ADA/WS/archiveFolder?src=macdc@C:\Tem
p&dst=Documentation@/Atempo%20Media%20Manager/V32SP3
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2011-09-13T15:53:13Z</request_date>
<operation>archiveFolder</operation>
<job_id>863</job_id>
<ReturnCode ADARetCode="1" />
<ErrorMsg />
</XmlResponse>
```

CHAPTER 8 - Copying Objects from a Platform to Another

To copy objects from a platform to another, apply this procedure.

Operation

`cp` copies a file or directory from a platform file system to another.

Options

These are the options that you can use with the `cp` command:

- `src=object_path` indicates the path and name of the object (directory or file) to copy. The syntax is different whether you copy local objects, or objects located on a NFS or CIFS share. See Syntax.
- `dst=directory_path` indicates the path and name of the directory in which you want to copy the object. The syntax is different whether you copy local objects, or objects located on a NFS or CIFS share. See Syntax.
- `dereference=1` indicates that Miria must follow the symbolic links and copy the target of the link.
- `no_dereference=1` indicates that Miria must copy only the symbolic link without following it.
- `debug=1` enables debug traces in the Miria Events.

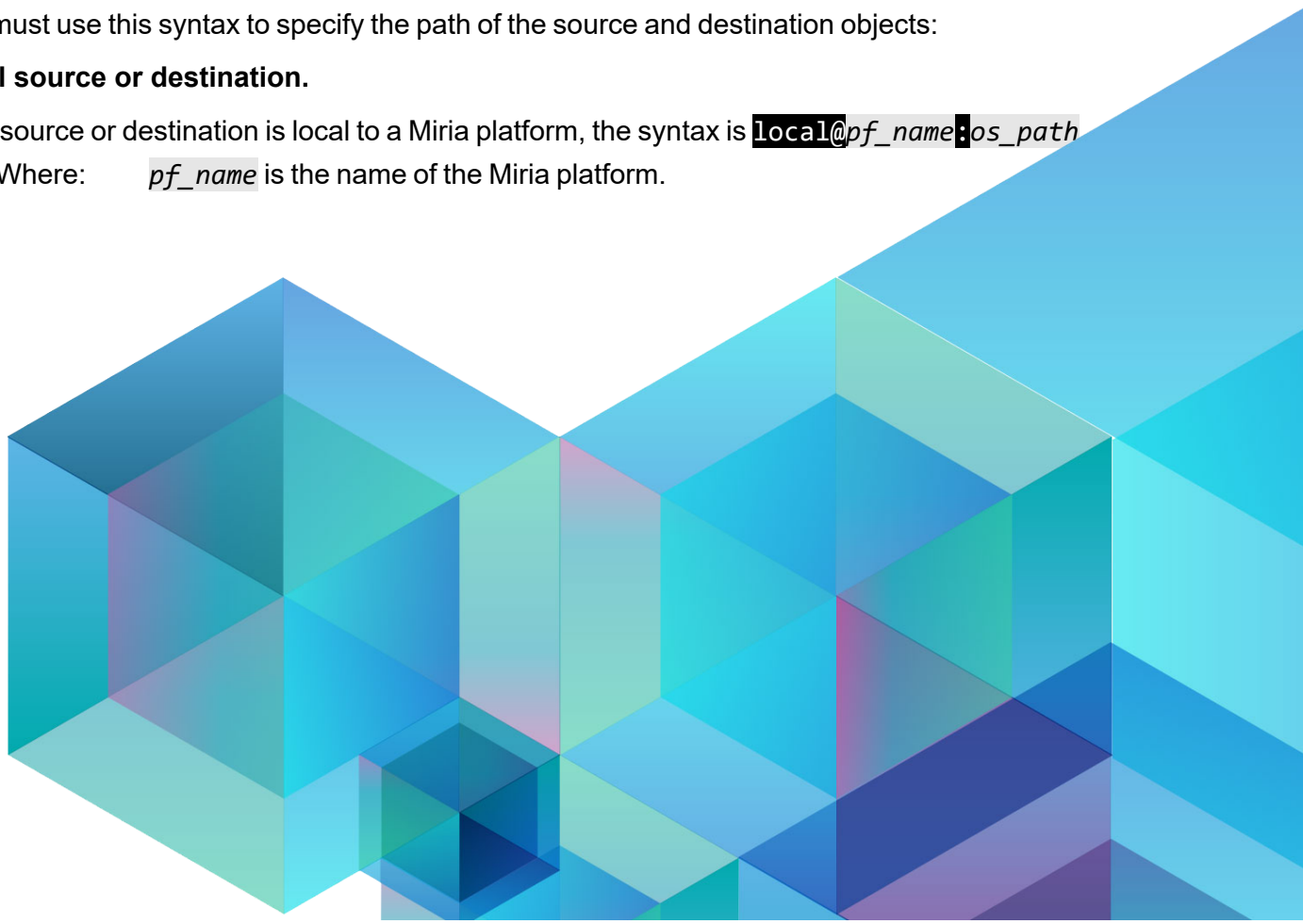
Syntax

You must use this syntax to specify the path of the source and destination objects:

Local source or destination.

If the source or destination is local to a Miria platform, the syntax is `local@`*`pf_name`*`:`*`os_path`*

Where: *`pf_name`* is the name of the Miria platform.



Where: *os_path* is the path of the source or destination.

Source or destination on a NFS or CIFS share.

The source or destination is located on a remote platform declared as a NAS in Miria, and accessible through NFS or CIFS.

- If the source or destination is accessible through an NFS share, the syntax is `nfs@agent_name:path`.
  Where: *agent_name* is the network name of the agent acting as gateway to the NAS.
  Where: *path* is the path of the source or destination on the mounting point on the agent.
- If the source or destination is accessible through CIFS share, the syntax is `cifs@unc_path`.
  Where: *unc_path* is the path of the source or destination in the UNC format (i.e., `\\NAS_name\share\path`).

HTTP Request

```
https://miria-server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/cp?src=local@dayos:c:\Temp\File.txt&dst=local@imacdoc:/Documents/bck
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2015-09-02T14:25:38Z</request_date>
<operation>cp</operation>
<job_id>2227</job_id>
<ReturnCode ADARetCode="1" />
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

The XML result provides the ID of the job without waiting for the job to complete. T
been successful, you must check with the `getJobStatus` web service or the Job List in the Administration Console.

CHAPTER 9 - Deleting an Archived File or Directory

To delete an archived file or directory, apply this procedure.

Operation

`deleteObject` deletes an archived file or directory from an archive folder.

Options

These are the options that you can use with the `deleteObject` command:

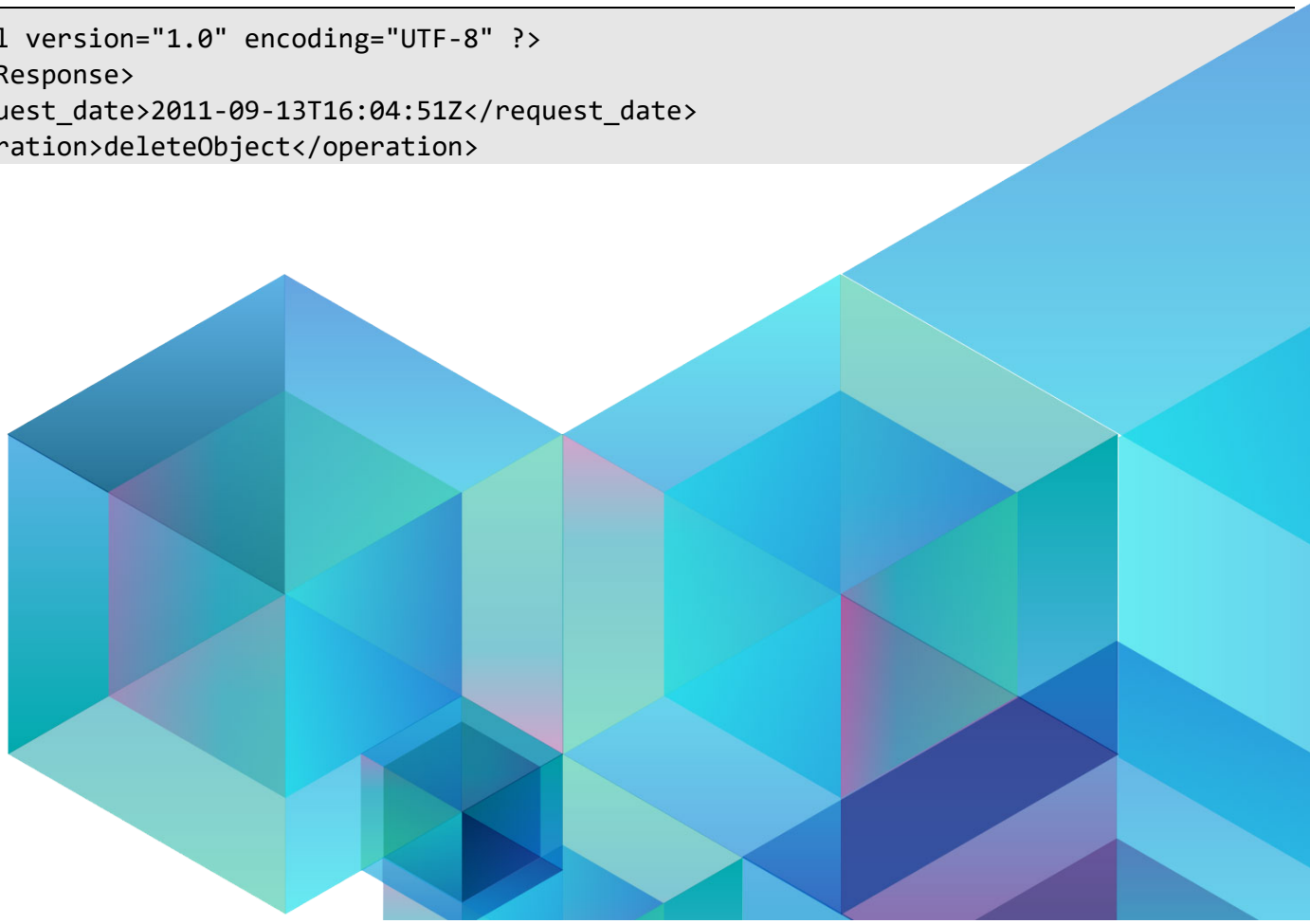
- `src=archive@path_in_archive` indicates the path of the archived file or directory that you want to delete.
- `debug=1` enables debug traces in the Events.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/deleteObject?src=Documen
tation@/Atempo%20Media%20Manager/V32SP3/ADA_UI.exe
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2011-09-13T16:04:51Z</request_date>
<operation>deleteObject</operation>
```



```
<ReturnCode ADARetCode="1" />
<ErrorMsg />
</XmlResponse>
```

CHAPTER 10 - Generating a Volume Report

To generate a volume report, apply this procedure.

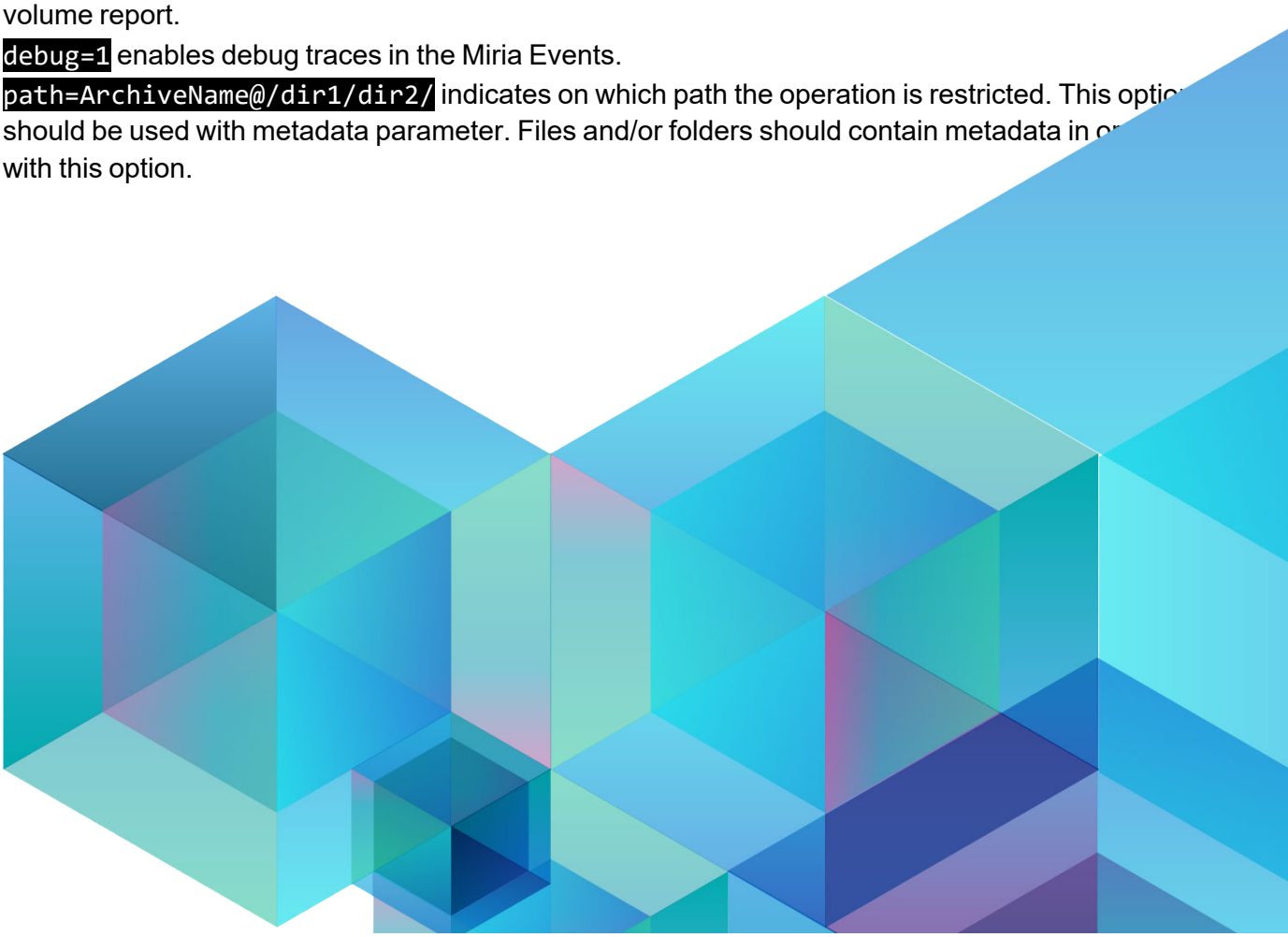
Operation

`reportVolume` generates an XML report that lists the number of objects archived and their volume, for each archive and each storage manager. If you do not specify any time limits, then the volume reported is the current volume. If you specify a start and/or end date, then the report lists the volume recorded every day of the interval at 23:55.

Options

These are the options that you can use with the `reportVolume` command:

- `start_date=aaammdd` indicates the starting date of the day by day report. You can use this option on its own or combined with the `end_date` and /or `days_interval` options. This option does not include the volume of the current day in the report.
- `end_date=aaammdd` indicates the end date of the day by day report. You can use this option on its own or combined with the `start_date` and /or `days_interval` options. This option does not include the volume of the current day in the report.
- `days_interval=value` indicates the frequency of the day by day report. By default, the report lists the volume recorded every day. For instance, set this option to 3 to list the volume every three days in the report.
- `with_archive_organization=1` displays the archive organizations in addition to the archives in the volume report.
- `debug=1` enables debug traces in the Miria Events.
- `path=ArchiveName@/dir1/dir2/` indicates on which path the operation is restricted. This optio
  should be used with metadata parameter. Files and/or folders should contain metadata in o
  with this option.



- `metadata=key:value,key:value,` … describes specific metadata the us files. This option could be combined with others options.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AE                                    b736531/ADA/WS/reportVolume?start_
date=2015/03/19&end_date=2015
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<ADA_GLOBAL_INFO Ada_Server_name="qaada-i500-w2k8R2-1" Date_generation="2015-03-
31T15:10:58Z" Ada_Server_id="22DCDD3E-3717F2F-5DC5C54-217AF62E" Hostname_
generation="qaada-i500-w2k8R2-1" xsi:noNamespaceSchemaLocation="http://qaada-i500-
w2k8R2-1:25422/xml/ADA_Report.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<ada_ReportDate ReportDate="2015-03-19T23:59:59Z">
<ada_Archive ada_Archive_Name="pax64">
<ada_Directory_Report Number="4"/>
<ada_File_Report Number="22" Size="57173805"/>
<ada_Folder_Report Number="1"/>
<ada_Link_Report Number="0"/>
<ada_Total_Report Number="27" Size="57173805"/>
</ada_Archive>
<ada_Archive ada_Archive_Name="pax128">
<ada_Directory_Report Number="22"/>
<ada_File_Report Number="48" Size="167772615"/>
<ada_Folder_Report Number="1"/>
<ada_Link_Report Number="0"/>
<ada_Total_Report Number="63" Size="167772615"/>
```

```
</ada_Archive>
<ada_Archive ada_Archive_Name="pax256">
<ada_Directory_Report Number="14"/>
<ada_File_Report Number="48" Size="167772615"/>
<ada_Folder_Report Number="1"/>
<ada_Link_Report Number="0"/>
<ada_Total_Report Number="63" Size="167772615"/>
</ada_Archive>
<ada_Archive ada_Archive_Name="1
<ada_Directory_Report Number="2"/>
<ada_File_Report Number="12" Size="29598640"/>
<ada_Folder_Report Number="1"/>
<ada_Link_Report Number="0"/>
<ada_Total_Report Number="15" Size="29598640"/>
</ada_Archive>
<ada_Storage ada_Storage_Type="Media Manager" ada_Storage_Name="amm">
<ada_Total_Report Number="185" Size="10541274316" MediaStreamSize="14296771072"
Media="9"/>
</ada_Storage>
<ada_Summary_Report>
<ada_Folder_Report Number="10"/>
<ada_Directory_Report Number="42"/>
<ada_Total_Report Number="237" Size="10541274316"/>
<ada_Archive_Report Number="4"/>
<ada_File_Report Number="185" Size="10541274316"/>
<ada_Link_Report Number="0"/>
<ada_Storage_Report Number="1"/>
</ada_Summary_Report>
</ada_ReportDate>
</ADA_GLOBAL_INFO></ada_ReportDate>
</ADA_GLOBAL_INFO>
```

See Field Description for a description of each field of the report.

CHAPTER 11 - Obtaining a Job Status

To obtain the status of a job, apply this procedure.

Operation

`getJobStatus` retrieves the status of a Miria job/sub-job, as well as its relations with other jobs/sub-jobs.

Options

These are the options that you can use with the `getJobStatus` command:

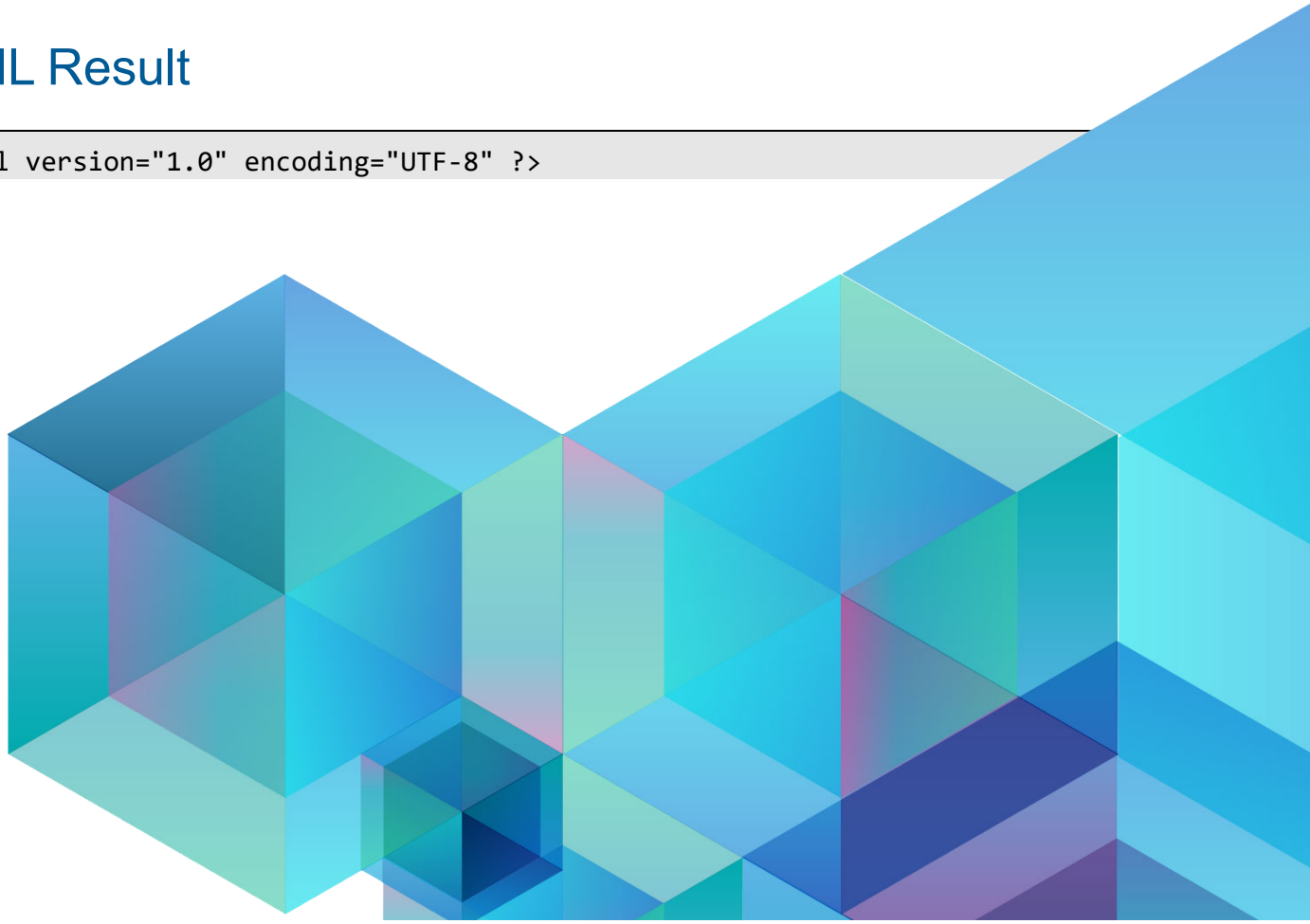
- `job_id=id`
- `subjob_id=id`
- `related_jobs=1`
  - With `job_id`, displays all the jobs issued from the same father sub-job.
  - With `subjob_id`, displays all the jobs that the sub-job has created.
- `main_subjob=1` displays the sub-job from which the specified job is issued.
- `debug=1` enables debug traces in the Events.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/getJobStatus?job_id=151
```

XML Result

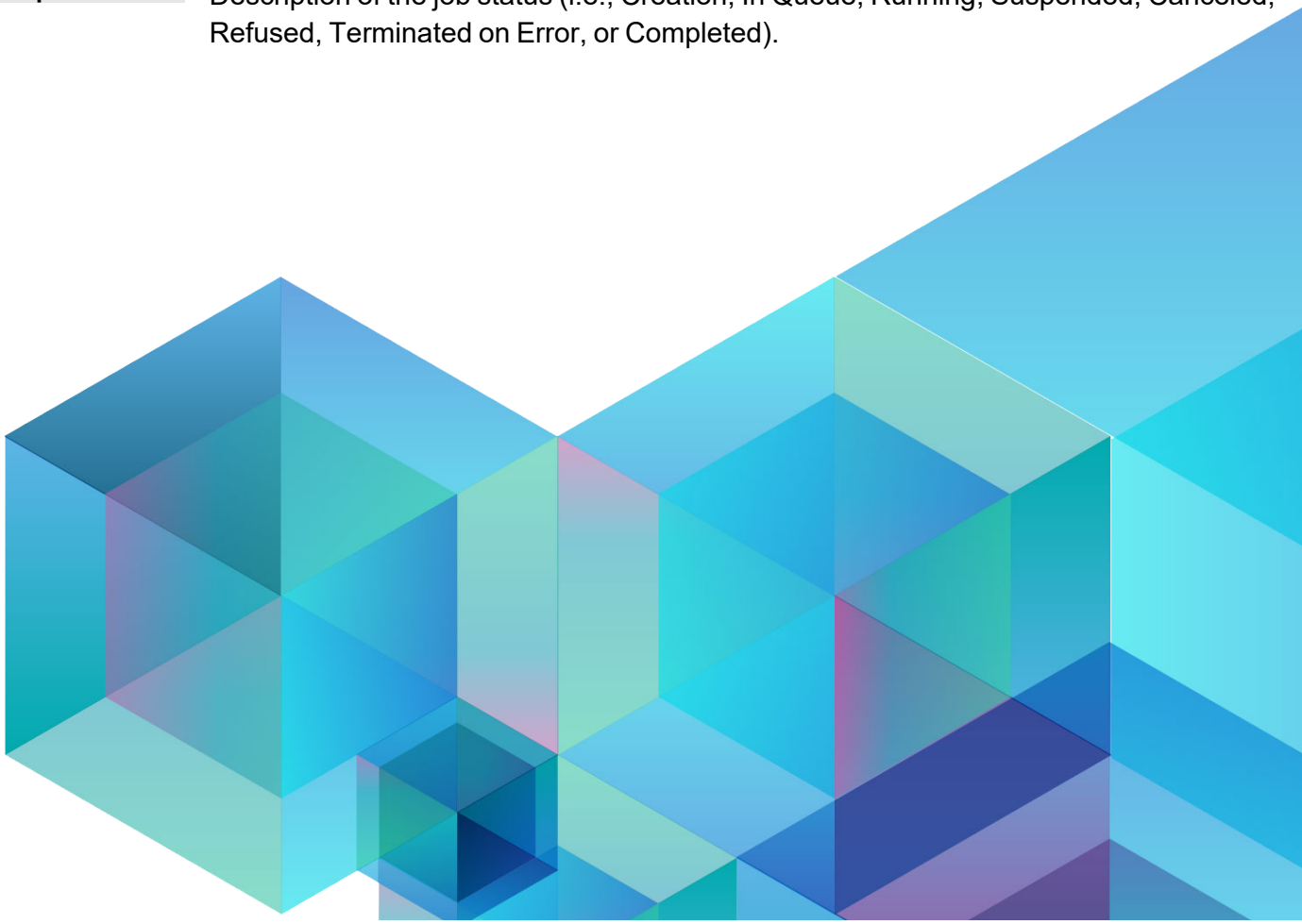
```
<?xml version="1.0" encoding="UTF-8" ?>
```



```
<XmlResponse>
<request_date>2016-06-15T15:16:50Z</request_date>
<operation>getJobStatus</operation>
<job_id>151</job_id>
<type>Archiving</type>
<status>8</status>
<description>Running</description>
<start_date>2020-05-20T15:21:
<end_date>2020-05-20T15:21:10
<durationTime>544</durationTime>
<priority>100</priority>
<progress>33.36</progress>
<subjobs>
<subjob subjob_id="204" subjob_type="Archiving" status="8" description="Running"
progress="33.36" total_rate="37985" throughput="37985" estimated_end="2016-02-
26T17:26:11Z" estimated_left="37985"/>
</subjobs>
<ReturnCode ADARetCode="1" />
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

This table describes the status information retrieved by the `getJobStatus` command:

Status Information	Description
type	Type of job (e.g., archiving, retrieval, retention).
status	Current status of the job - internal reference.
description	Description of the job status (i.e., Creation, In Queue, Running, Suspended, Canceled, Refused, Terminated on Error, or Completed).



Status Information	Description
`priority`	Internal priority level of a job. The values can be 0 (low priority), 50 (normal), or 100 (high priority).
`progress`	Job progress in percentage.
`subjob_id`	ID of the related sub-job.
`subjob_type`	Type of sub-job (e.g., archiving, retrieval, retention).
`total_rate`	Average rate of the data flow in bytes per second from the job beginning up to the current time.
`throughput`	Rate of the data flow in bytes per second at the current time.
`estimated_end`	Date and time at which Miria estimates that the sub-job will end.
`estimated_left`	Estimation of the number of seconds remaining before the sub job completes.

For details on job statuses, see .

CHAPTER 12 - Obtaining a Drive Status

To obtain the status of a drive, apply this procedure.

Operation

`getDriveStatus` retrieves the status of Media Manager drives.

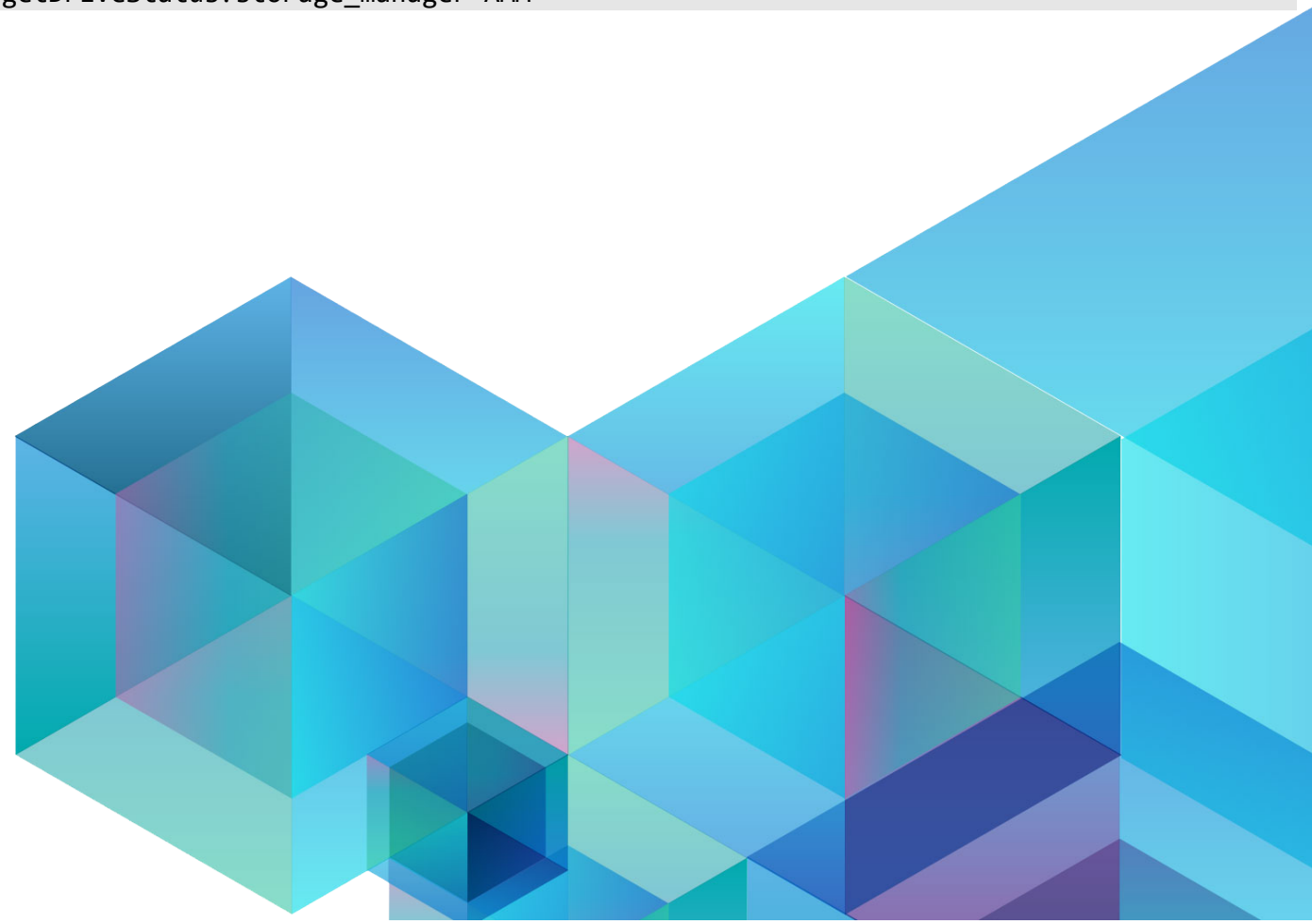
Options

These are the options that you can use with the `getDriveStatus` command:

- `drive_name=name` indicates the name of a particular drive for which you want to obtain the status.
- `library_name=name` indicates the name of a particular library for which you want to obtain the drive status.
- `storage_manager=name` indicates the name of the Media Manager storage manager for which you want to obtain the drive status.
- `display_name=1` returns the alias of the name of the drive, instead of its usual name.
- `alias=1` returns the alias of the drive in addition to its name.
- `connection=1` returns drive connection information.
- `debug=1` enables debug traces in the Events.

HTTP Request

```
https://miria-server/meta/D5848F8CC996DD540CC5C020514F5C60/721b736531/ADA_Vault_Svc/getDriveStatus?storage_manager=AMM
```



XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<ReturnCode ADARetCode="1" />
<ErrorMsg />
<request_date>2015-04-29T14:3[illegible]</request_date>
<operation>getDriveStatus</operation>
<drive name="DOCL700_D2" media="" connection_status="online" drive_status="enabled"
allocated="no" library="DOC_L700" />
<drive name="DOCL700_D1" media="" connection_status="online" drive_status="enabled"
allocated="no" library="DOC_L700" />
</XmlResponse>
```

This table describes the status information retrieved by the `getDriveStatus` command:

Status Information	Description
`drive name`	Name of the drive handled by Media Manager. By default, it is the serial number of the drive for a tape drive.
`drive_alias`	Alias of the drive, if any.
`media`	Barcode of the media located in the drive, if any.
`connection_status`	Media Manager connection status of the drive: `online` or `offline`.

Status Information	Description
`drive_status`	Status of the drive. These are the valid values: • `enabled` • `disabled` • `unknow` • `mainte ar e`
`connection`	Number of platforms configured to use the drive.
`connection_active`	Number of active platforms configured to use the drive.
`allocated`	Assignment status of the drive: • `yes` (allocated by a host). • `no` (not allocated).
`library`	Name of the library where the drives are located.
`library_alias`	Alias of the library, if any.

CHAPTER 13 - Obtaining a Library Status

To obtain the status of a library, apply this procedure.

Operation

`getLibraryStatus` retrieves the status of the Media Manager libraries.

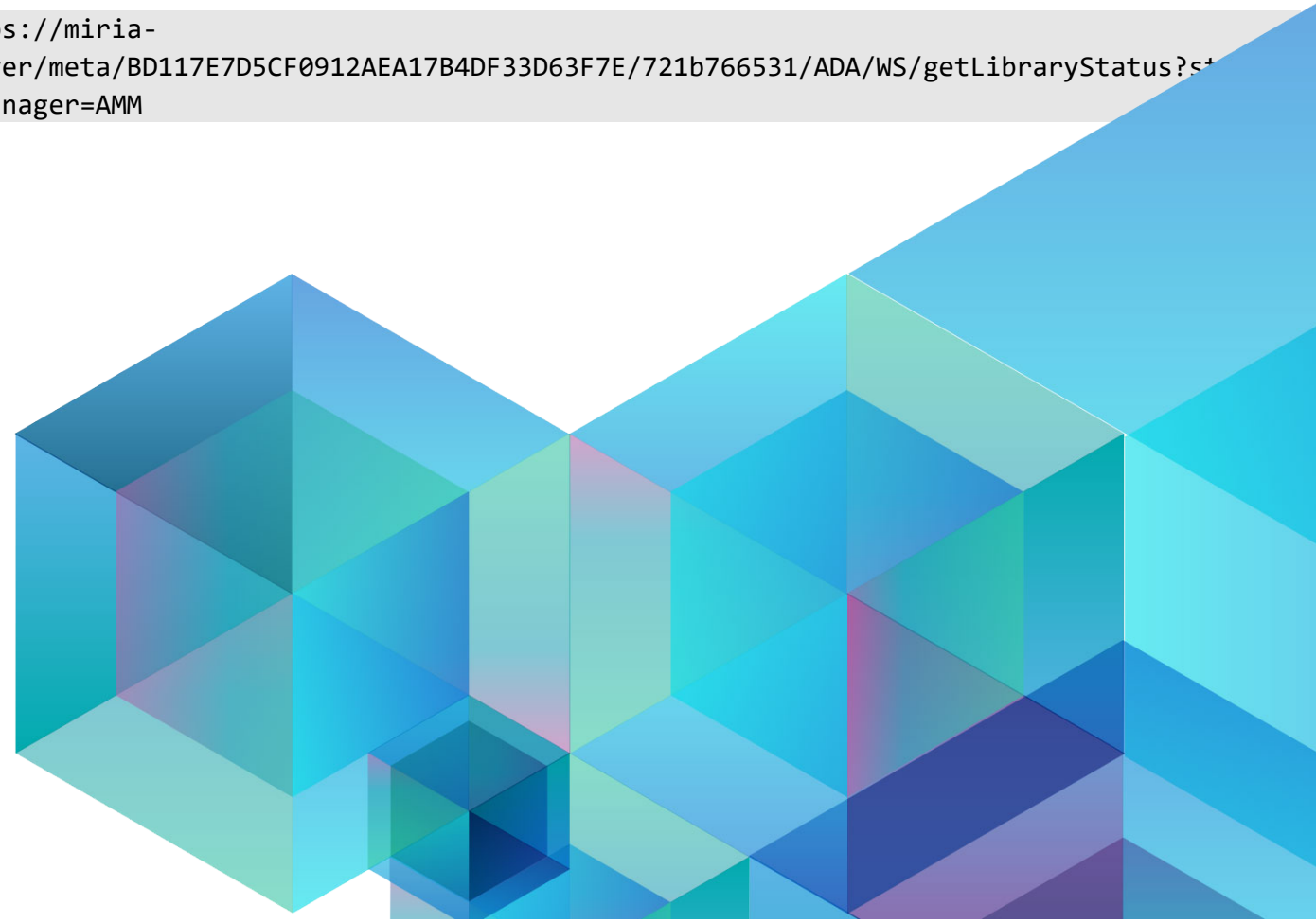
Options

These are the options that you can use with the `getLibraryStatus` command:

- `library_name=name` indicates the name of a particular library for which you want to obtain the status. Either this option or the `application_name` option is required.
- `application_name=name` indicates the name of a particular Media Manager application for which you want to obtain the library status. Either this option or the `library_name` option is required.
- `storage_manager=name` indicates the name of the Media Manager storage manager for which you want to obtain the library status.
- `display_name=1` returns the alias of the name of the library, instead of its usual name.
- `alias=1` returns the alias of the library in addition to its name.
- `force_offline_stat=1` always displays information about offline media, even if there is none.
- `debug=1` enables debug traces in the Miria Events.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b766531/ADA/WS/getLibraryStatus?st
e_manager=AMM
```

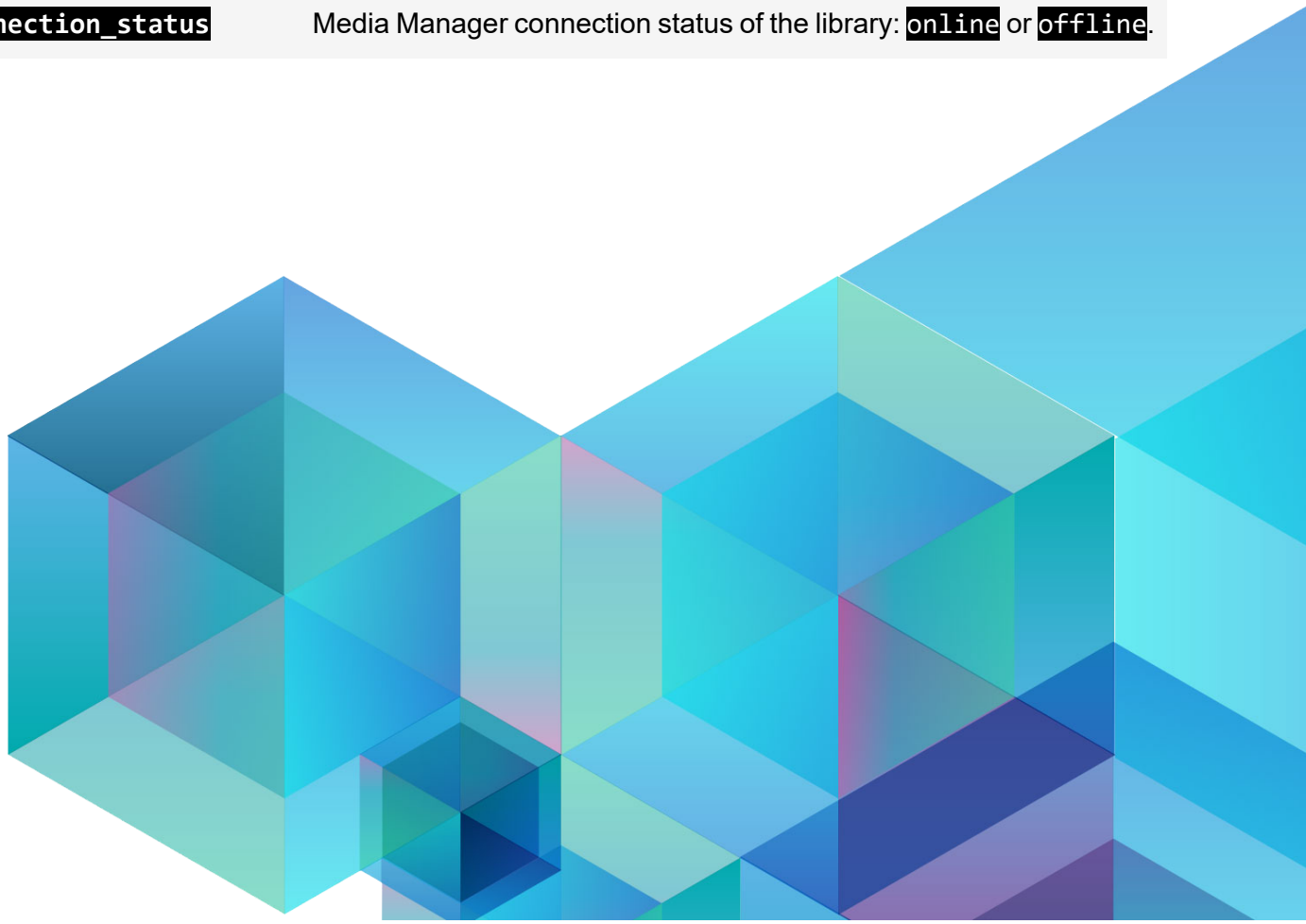


XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2015-04-16T16:57:06Z</request_date>
<operation>getLibraryStatus</operation>
<offline>
<media_number_by_status others="0" blank="0" scratch="0" orphan="0" assigned="2"
unknown="0" />
</offline>
<library name="DOC_L700" type="scsi" connection_status="online" library_
status="enabled" slot_number="12" free_slot_number="2">
<media_number_by_status incompatible="0" others="0" blank="0" scratch="0" orphan="0"
unknown="0" assigned="10" />
</library>
<ReturnCode ADARetCode="1" />
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

This table describes the status information retrieved by the `getLibraryStatus` command:

Status Information	Description
`library name`	Name of the library handled by Media Manager.
`type`	Library type: SCSI or ACSLS.
`library_alias`	Alias of the library, if any.
`connection_status`	Media Manager connection status of the library: `online` or `offline`.



Status Information	Description
`library_status`	Status of the library. These are the valid values: • `enabled` • `disabled` • `unknown` • `maintenance`
`slot_number`	Total number of slots in the library.
`free_slot_number`	Number of free slots in the library.
`offline`	Number of offline media and their statuses.
`media_number_by_status`	Number of media having these statuses. These are the valid values: • Assigned • Scratch • Blank • Unknown • Orphan • Others

CHAPTER 14 - Obtaining the List of Devices

To obtain the List of devices, apply this procedure.

Operation

`getDeviceList` retrieves the list of devices connected to a particular platform.

Options

These are the options that you can use with the `getDeviceList` command:

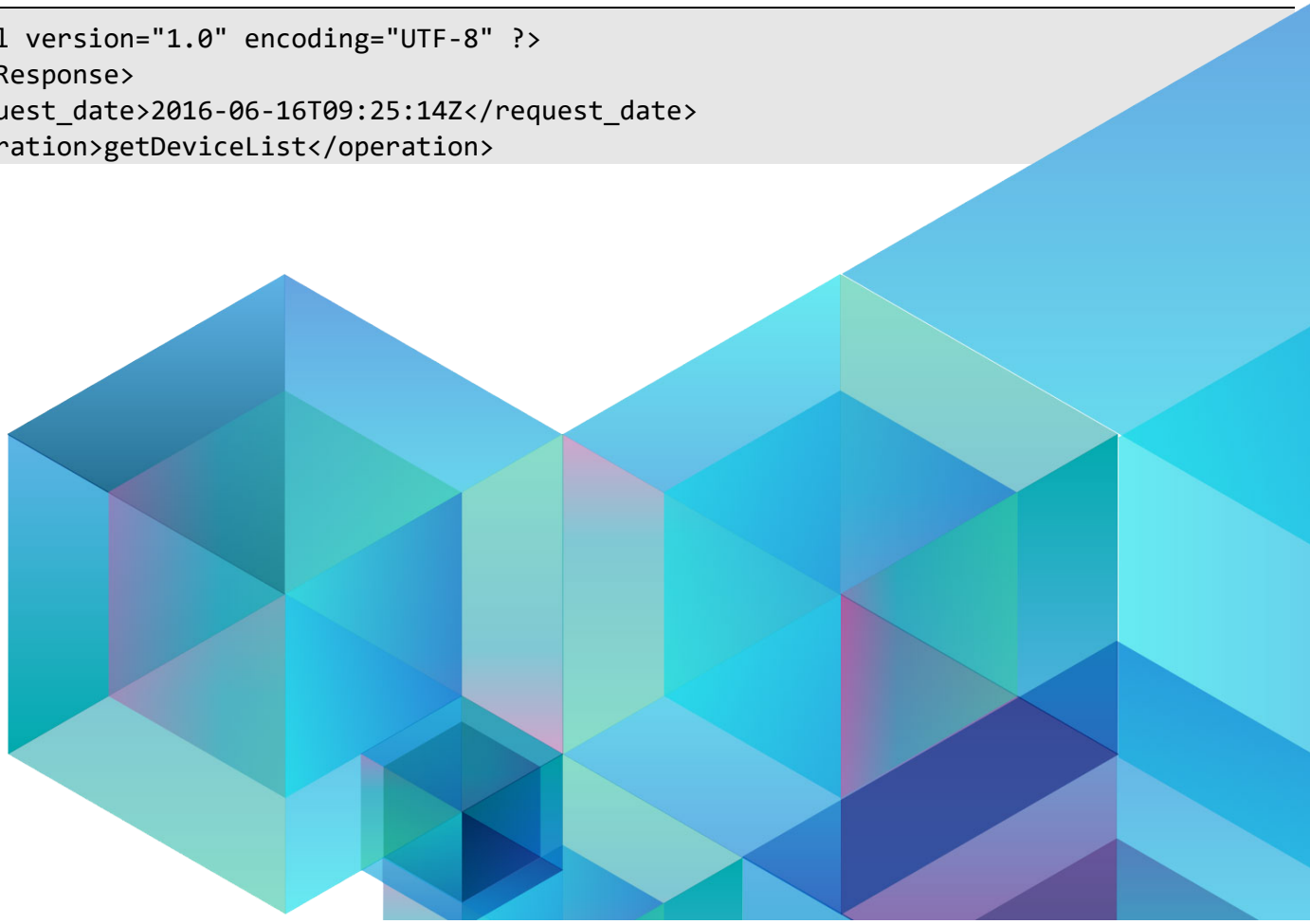
- `platform_name` indicates the name of a particular platform for which you want to obtain the list of devices.
- `debug=1` enables debug traces in the Events.

HTTP Request

```
https://miria-
server/meta/26887AE88F04F9A6991AB86DC50861A6/721b736531/ADA/WS/getDeviceList?platform_
name=frlul8adadoc
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2016-06-16T09:25:14Z</request_date>
<operation>getDeviceList</operation>
```



```
<device name="559000203260" type="Library" alias="559000203260" on
OS, configured in AMM" LM_status="Enabled" connection_status="Enabled" serial_
number="559000203260" scsi_vendor="STK" scsi_product="SL500" scsi_revision="1373"
device_descriptor="spt_c3b0t3l0"/>
<device name="1068016668" type="Drive" alias="1068016668" scan_status="Present on OS,
configured in AMM" DM_status="Enabled" connection_status="Online" serial_
number="1068016668" scsi_vendor="IBM" scsi_product="ULT3580-HH5" scsi_revision="D2AD"
device_descriptor="c2b0t3l0"/>
<device name="1068017011" type="Drive" alias="1068017011" scan_status="Present on OS,
configured in AMM" DM_status="Enabled" connection_status="Online" serial_
number="1068017011" scsi_vendor="IBM" scsi_product="ULT3580-HH5" scsi_revision="D2AD"
device_descriptor="c2b0t4l0"/>
<device name="1013004C53" type="Drive" alias="1013004C53" scan_status="Present on OS"
serial_number="1013004C53" scsi_vendor="IBM" scsi_product="ULTRIUM-HH7" scsi_
revision="G341" device_descriptor="c2b0t5l0"/>
<ReturnCode ADARetCode="1" />
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

This table describes the status information retrieved by the `getDeviceList` command:

Information	Description
device name	Name of the library or drive connected to the platform.
type	Type of device: Library or Drive.
alias	Alias of the device, if any.

Information	Description
`scan_status`	Scan status of the device. These are the possible values : • Present on OS, configured in AMM. • Configured in AMM, no longer present on OS. • Present on OS. The device is present on the OS but not configured in Media Manager. • Present on OS, configured in AMM, drive not attached to library. • Configured in AMM. The device is configured in Media Manager, it does not have to be present at the OS (i.e., ACSLS).
`LM_status`	Status of the library. These are the possible values: • `enabled` • `disabled` • `unknown` • `maintenance`
`DM_status`	Status of the drive. These are the possible values: • `enabled` • `disabled` • `unknown` • `maintenance`
`connection_status`	Media Manager connection status of the device: `online` or `offline`.
`library_name`	Name of the library in which the drive is located.
`drive_address`	Location of the drive in the library.
`serial_number`	Serial number of the drive or library.

Information	Description
`scsi_vendor`	Name of the vendor of the drive or library.
`scsi_product`	Name of the model of the drive or library.
`scsi_revision`	Revision level of the drive or library firmware.
`device_descriptor`	SCSI addr[illegible]y.

CHAPTER 15 - Obtaining the List of Media

To obtain the List of media, apply this procedure.

Operation

`getMediaList` retrieves the list of media based on some criteria.

Options

These are the options that you can use with the `getMediaList` command:

- `all_media=1` lists all the media present in the Miria database.
- `allocated=1` lists the media in use by a Media Manager application.
- `amm_drive_class=name1[,name2,name3,...]` lists the media located in Media Manager drives of this class. Use the `ada_amm -drive_type` command to display the list of available drive classes (e.g., ODA_1, LTO-ULTRIUM, etc). See Obtaining Information and Acting on Devices.
- `amm_drive_techno=name1[,name2,name3,...]` lists the media located in Media Manager drives of this technology. Use the `ada_amm -drive_type` command to display the list of available drive technologies (i.e., SCSI, ODA, DISK). See Obtaining Information and Acting on Devices.
- `amm_drive_type=name1[,name2,name3,...]` lists the media located in Media Manager drives of this type. Use the `ada_amm -drive_type` command to display the list of available drive types (e.g., ODA_D55U, LTO-5, etc.). See Obtaining Information and Acting on Devices.
- `amm_library_name=name` lists the media located in this Media Manager library.
- `amm_media_status=assigned|blank|orphan|others|scratch|undefined`: lists the media having this status.
- `amm_media_status_detail=duplicate_digest|media_incompatible|none|read_error|scratch_error`: lists the media of type `other` having this status detail.

- `amm_media_type=name1[,name2,name3,...]` lists the media with this typ
  `media_type` command to display the list of available media types (e.g., ODA-300GB, IBM-03592, T10KA, etc). See Obtaining Information and Acting on Devices.
- `app_name=name` indicates the name of the Media Manager application for which you want to obtain the media list.
- `archive_name=name` indicates the name of the project archive for which you want to obtain the media list.
- `archive_name_rule=name` indicates the name of the project archive of the media rule for which you want to obtain the media list.
- `archive_org=name` indicates the name of the archive organization for which you want to obtain the media list.
- `archive_org_rule=name` indicates the name of the archive organization of the media rule for which you want to obtain the media list.
- `archive_rule=1` lists all media belonging to a media rule *By Archive*.
- `barcode=name1[,name2,name3,...]` indicates the barcode(s) of the media(s) that you want to list.
- `cause=none|full|on_error|on_write_error|on_read_error|on_spanning|on_check_integrity|on_discover` indicates the error cause of the media for which you want to obtain the media list.
- `custom_name_rule=name` indicates the name of the custom media rule for which you want to obtain the media list.
- `custom_rule=1` lists all media belonging to any custom media rule.
- `data_partition=value` indicates the value of the LTFS data partition for which you want to obtain the media list. The value can be `0` or `1`.
- `in_drive=name` indicates the name of the drive in which the media to list is located.
- `job_id=id` indicates the ID of the job for which you want to obtain the media list.
- `job_id_rule=id` indicates the ID of the media rule *By Job* for which you want to obtain the media list.
- `job_rule=1` lists all media belonging to a media rule *By Job*.
- `ltfs_owner=name` indicates the name of the LTFS owner for which you want to obtain the media list. Available only for LTFS media.
- `ltfs_volume=name` indicates the name of the LTFS volume for which you want to obtain the medi
  Available only for LTFS media.

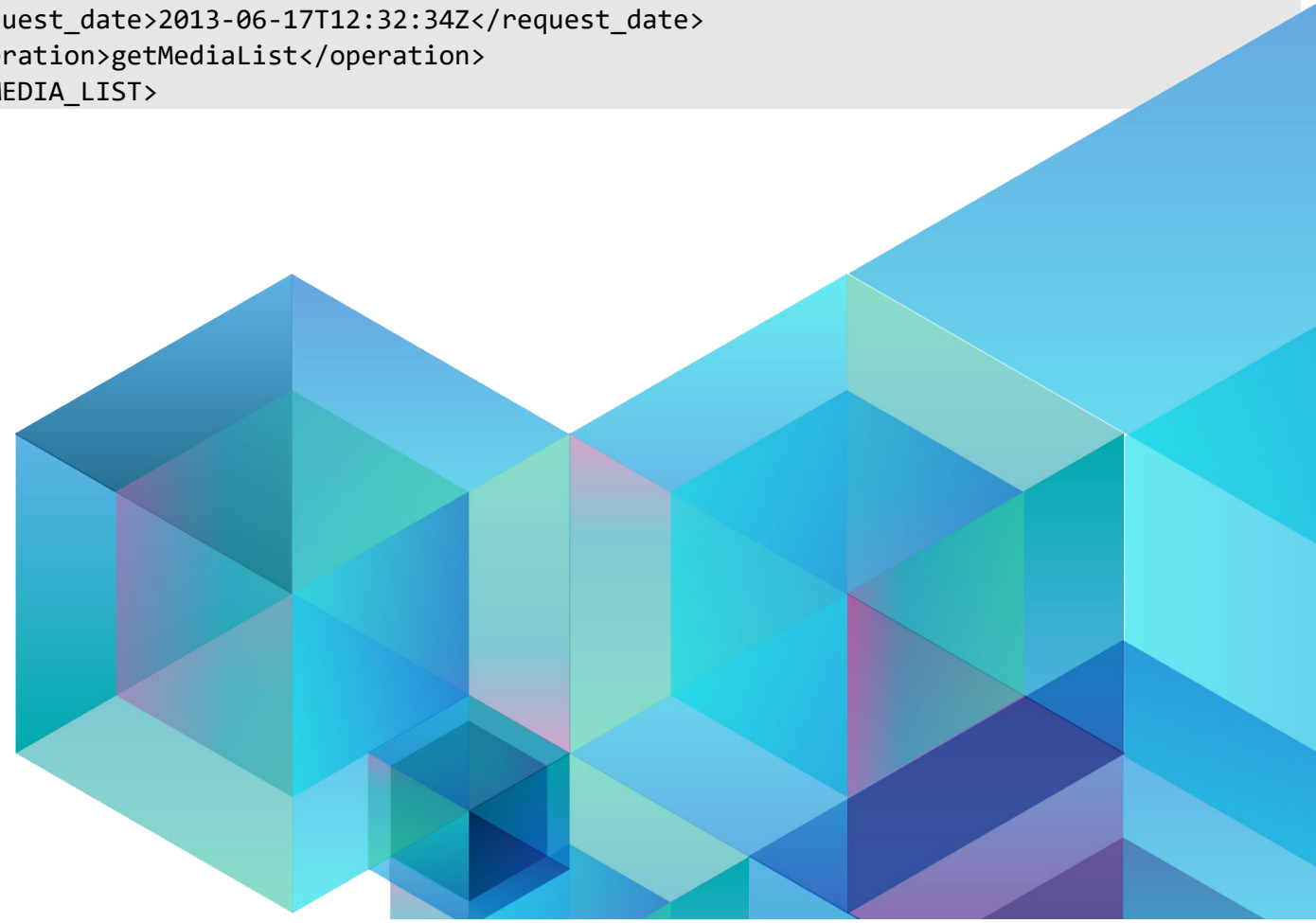
- `media_format=pax|tar|tina|cpio|sidf|ltfs`: Format of the media for the media list.
- `mounted=1` lists the media mounted in any drive.
- `offline=1` lists the offline media.
- `online=1` lists the media located in a library.
- `sm_name=name` indicates the name of the storage manager for which you want to obtain the media list.
- `smc_name=name` indicates the name of the storage manager container for which you want to obtain the media list.
- `status=new|open|closed|suspended|empty` indicates the Status of the media for which you want to obtain the media list.
- `subjob_id=id` indicates the ID of the sub-job for which you want to obtain the media list.
- `user_name=name` indicates the name of the user archive for which you want to obtain the media list.
- `user_name_rule=name` indicates the name of the user archive of the media rule for which you want to obtain the media list.
- `debug=1` enables debug traces in the Miria Events.

HTTP Request

```
https://miria-server/meta/26887AE88F04F9A6991AB86DC50861A6/721b736531/ADA/WS/getMediaList?status=closed
```

XML Result

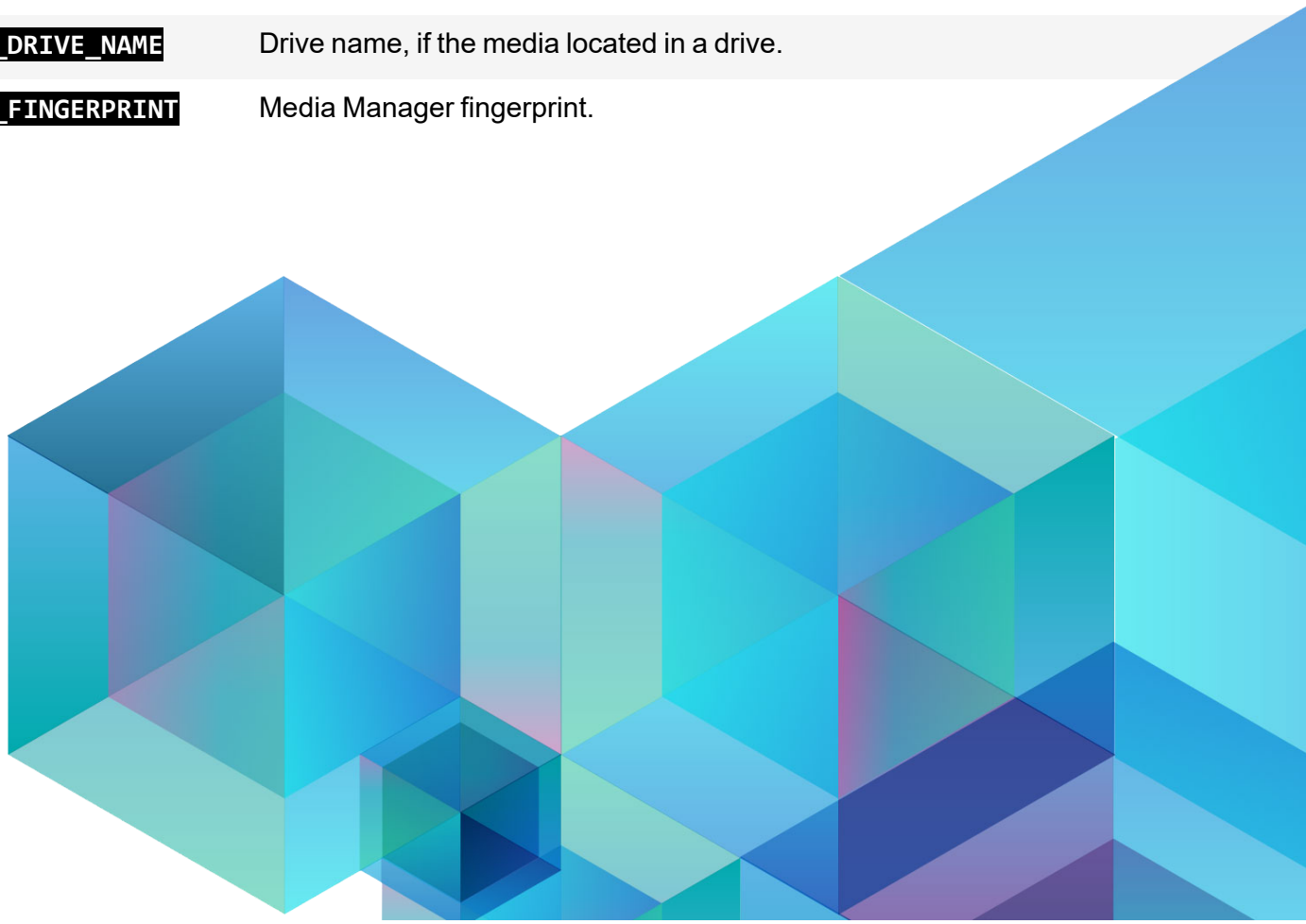
```
<?xml version="1.0" encoding="UTF-8" ?>
- <XmlResponse>
<request_date>2013-06-17T12:32:34Z</request_date>
<operation>getMediaList</operation>
- <MEDIA_LIST>
```



```
<MEDIA AMM_ALLOCATION="0" AMM_CARTRIDGE_STATE="0" AMM_CARTRIDGE_
NAME="DOCL700_D1" AMM_FINGERPRINT="d4Ra4BX5YRt8B36/ISKLgw==" AMM_LIBRARY_NAME="DOC_
L700" AMM_MEDIA_GROUP="cartgrp_DOC_L700" AMM_MOUNTED="1" AMM_ONLINE="1" AMM_PREVENT_
EJECT="no" AMM_SLOT_HOME="d0" MD_BLOCKSIZE="65536" MD_CAUSE="0" MD_
FINGERPRINT="d4Ra4BX5YRt8B36/ISKLgw==" MD_FORMAT="0" MD_LPOS="381" MD_NAME="B00005L4"
MD_NODE="1749" MD_REMAINING_VOLUME="293181496" MD_STATUS="2" MD_TAPEFILE="3" MD_
TYPE="2" MD_VOLUME="24772608" MR_TYPE="0" MC_NODE="15" SM_NODE="27" />
</MEDIA_LIST>
<ReturnCode ADARetCode="1" />
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

This table describes the information retrieved for each media by the `getMediaList` command:

Information	Description
`AMM_ALLOCATION`	Indicates whether an application has reserved the media. These are the valid values: • `0` indicates that the media is not reserved. • `1` indicates that the media is reserved.
`AMM_CARTRIDGE_STATE`	Indicates the Media Manager status of the media. These are the valid values: • `0` (assigned) • `1` (scratch) • `2` (undefined) • `3` (orphan) • `99` (other)
`AMM_CARTRIDGE_TYPE`	Type of media (e.g., `LTO`).
`AMM_DRIVE_NAME`	Drive name, if the media located in a drive.
`AMM_FINGERPRINT`	Media Manager fingerprint.



Information	Description
AMM_LIBRARY_NAME	Name of the library in which the media is located.
AMM_MEDIA_GROUP	Media group to which the media belongs.
AMM_MOUNTED	0 indicates that the media is not mounted. 1 indicates that the media is mounted in a drive.
AMM_ONLINE	0 indicates that the media is offline. 1 indicates that the media is online in the library.
AMM_PREVENT_EJECT	0 indicates that the library allows media ejection. 1 indicates that the library prevents media from ejection.
AMM_SLOT_HOME	Slot of the library in which the media is located at the time of the request.
MD_CAUSE	Error cause of the media.
MD_FORMAT	Tape format of the media.
MD_LPOS	Logical position of the media if it is located in a drive.
MD_NAME	Barcode of the media.
MD_NODE	Internal identifier of the media in the Miria database.
MD_REMAINING_VOLUME	Volume remaining on the media in bytes.

Information	Description
`MD_STATUS`	Indicates the Miria status of the media. These are the valid values: • `0` (new) • `1` (open) • `2` (closed) • `3` (suspended) • `4` (empty) • `5` (unrecoverable)
`MD_TAPEFILE`	Number of tape files on the media.
`MD_TYPE`	Media type. These are the valid values: • `1` (file) • `2` (tape)
`MD_VOLUME`	Volume used in bytes.
`MR_TYPE`	Media Rule Type. These are the valid values: • `0` (none) • `1` (by archive) • `2` (custom) • `3` (by job)
`SMC_NODE`	Internal identifier of the storage manager container in Miria database.
`SM_NODE`	Internal identifier of the storage manager in Miria database.

CHAPTER 16 - Obtaining a Platform Status

To obtain the status of a platform, apply this procedure.

Operation

`getPlatformStatus` retrieves the status, type and operating system of a Miria platform.

Options

These are the options that you can use with the `getPlatformStatus` command:

- `platform_name` indicates the name of a particular platform for which you want to obtain the status.
- `snapshot_list` gets the list of snapshots.
- `debug=1` enables debug traces in the Events.

HTTP Request

```
https://miria-
server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b766531/ADA/WS/getPlatformStatus
```

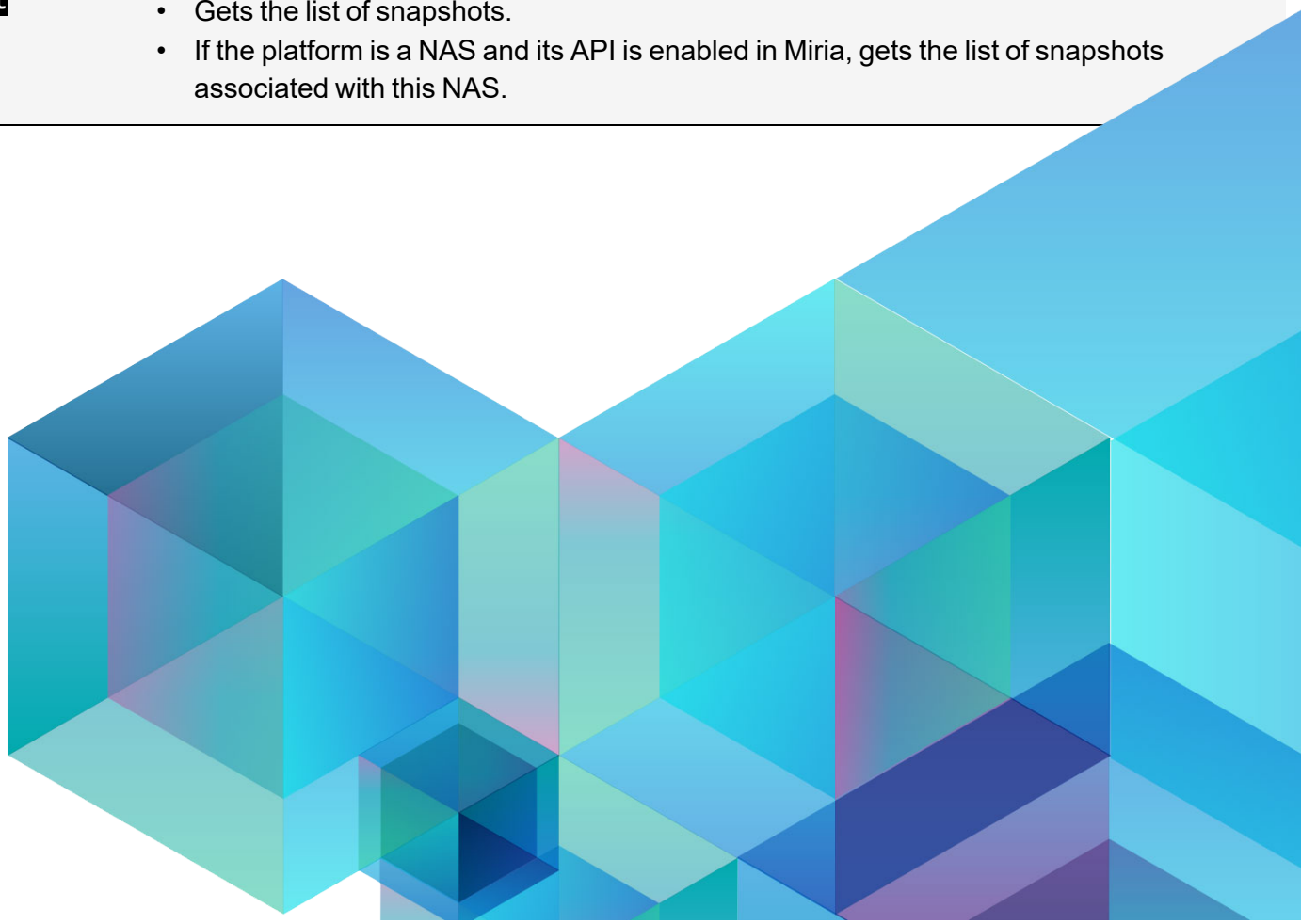
XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2015-04-16T15:55:49Z</request_date>
<operation>getPlatformStatus</operation>
<platform type="agent" name="adadoc" status="enable" os_type="Windows" />
```

```
<platform type="agent" name="imacdoc" status="enable" os_type="U
<platform type="nas" name="adashare" status="enable" agent_windows= adadoc  nas_
type="Avid Unity" />
<platform type="pool" name="PPool_1" status="disable" os_type="" />
<platform type="pool" name="PPool_2" status="disable" os_type="" />
<ReturnCode ADARetCode="1" />
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
</XmlResponse>
```

This table describes the status information retrieved for each platform by the `getPlatformStatus` command:

Information	Description
`name`	Platform name.
`type`	Platform type: agent or nas.
`status`	Platform status: enabled or disabled.
`os_type`	Type of the platform operating system if the platform is of agent type.
`agent_windows`	Name of the Windows agent if the platform is of NAS type.
`agent_unix_mac`	Name of the Unix or macOS agent if the platform is of NAS type.
`nas_type`	Type of the NAS.
`snapshot_list`	When set to `1`, the `snapshot_list` option obtains these results: • Gets the list of snapshots. • If the platform is a NAS and its API is enabled in Miria, gets the list of snapshots associated with this NAS.



CHAPTER 17 - Obtaining an Archived File Status

To obtain the status of an archived file, apply this procedure.

Operation

`getFileStatus` retrieves the status of the last instance of an archived file.

Options

These are the options that you can use with the `getFileStatus` command:

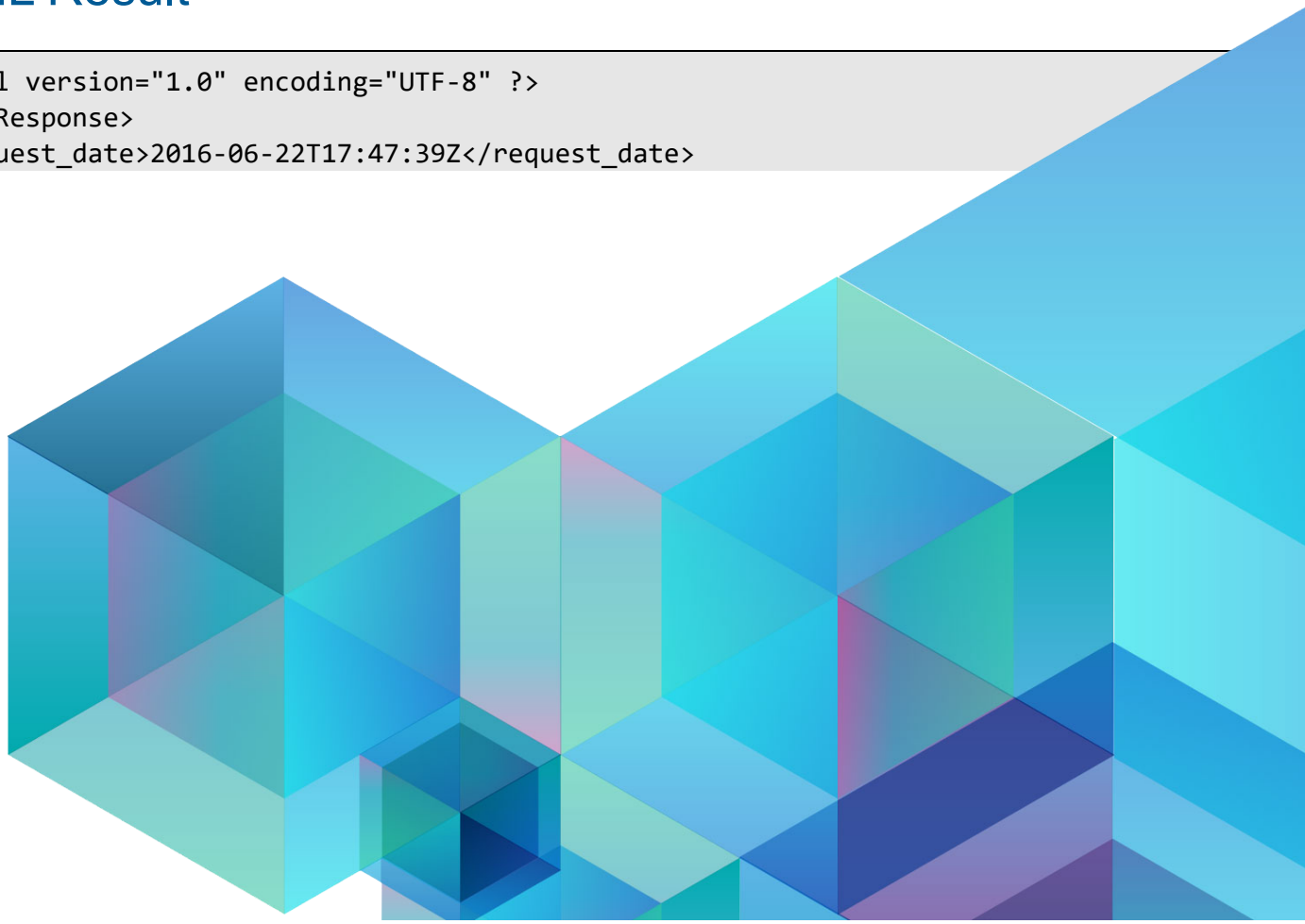
- `src=archive@path_in_archive` indicates the archived file path and name for which you want to obtain the status.
- `pr_info=1` gets the file duration if the specified archived file is a media asset.
- `debug=1` enables debug traces in the Events.

HTTP Request

```
https://miria-
server/meta/26887AE88F04F9A6991AB86DC50861A6/721b736531/ADA/WS/getFileStatus?src=amm@/0
001/small_shaak.mov&pr_info=1
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2016-06-22T17:47:39Z</request_date>
```



```
<operation>getFileStatus</operation>
<filename>amm@/0001/small_shaak.mov</filename>
<instance owner_group="root" owner_user="root" file_size="2914349" archiving_
date="2016/06/21-16:46:31" original_location="/DT/ADA/SOURCE/small_shaak.mov" mime_
type="" creation_time="2015/04/17-16:46:10" last_update="2015/04/17-17:27:20" last_
access="2015/04/17-16:46:26" duration="00:00:23:266 msec"/>
<storage>
<storage_manager name="amm" type="AMM_Storage_Manager">
<media_list>
<media name="SNW473L2" allocation="no" online="yes"/>
</media_list>
</storage_manager>
</storage>
<ReturnCode ADARetCode="1"/>
<ErrorMsg></ErrorMsg>
</XmlResponse>
```

This table describes the status information retrieved by the `getFileStatus` command:

Status Information	Description
`filename`	Archived file name and path.

Status Information	Description
`instance`	Information on the archived file last instance: • `owner_group` indicates the group to which the owner of the archived file belongs. • `owner_user` indicates the owner of the archived file. • `file_size` indicates the size of the archived file in bytes. • `archiving_date` indicates the date when the file was last archived. • `original_location` indicates the original path of the archived file on the file system. • `mime_type` indicates the MIME type of the archived file. • `creation_time` indicates the date when the file was created on the file system. • `last_update` indicates the date when the file was last updated. • `last_access` indicates the date when the file was last accessed. • `duration` indicates a media asset duration. This information is only available if the `pr_info` option is enabled and if the file is a media asset.
`storage_manager`	Information on the storage managers used to archive the file: • `name` indicates the storage manager name. • `type` indicates the storage manager type.

Status Information	Description
`media_list`	To be able to view media list information, you must set the View Media List advanced setting to `Yes`. List of the media on which the file is archived when using a Media Manager storage manager: • `name` indicates the name of the media. • `allocation` indicates whether the media is mounted. These are the valid values: – `Yes` – `No` • `online` indicates whether the media is online. These are the valid values: – `Yes`. The media is in the library. – `No`. The media is offline.

CHAPTER 18 - Obtaining the Contents of an Archive Folder

To obtain the contents of an archive folder, apply this procedure.

Operation

`ls` lists the contents of an archive folder.

Options

These are the options that you can use with the `ls` command:

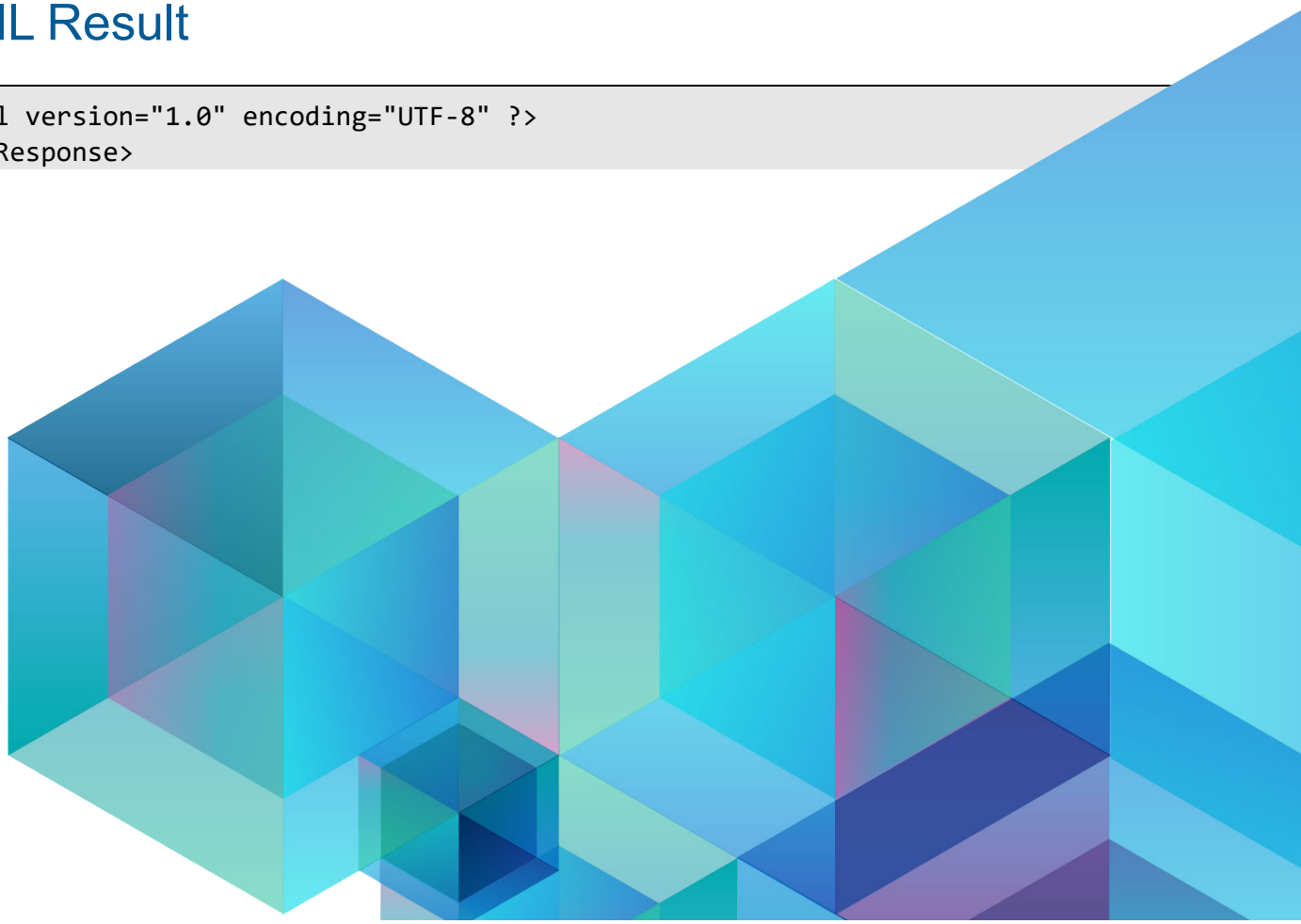
- `src=`*`archive@path_in_archive`* indicates the path and name of the archive folder for which you want to list the contents.
- `debug=1` enables debug traces in the Events.

HTTP Request

```
https://miria-server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/ls?src=Documentation@/Atempo%20Digital%20Archive
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
```



```
<request_date>2011-09-13T16:19:05Z</request_date>
<operation>ls</operation>
<object name="GIOTTO_Ascension_-Capell.jpg" type="file" />
<object name="Movies" type="directory" />
<object name="Screen" type="directory" />
<object name="Setup" type="directory" />
<ReturnCode ADARetCode="1" />
<ErrorMsg />
</XmlResponse>
```

CHAPTER 19 - Obtaining Server Information

To get information about the Miria server, apply this procedure.

Operation

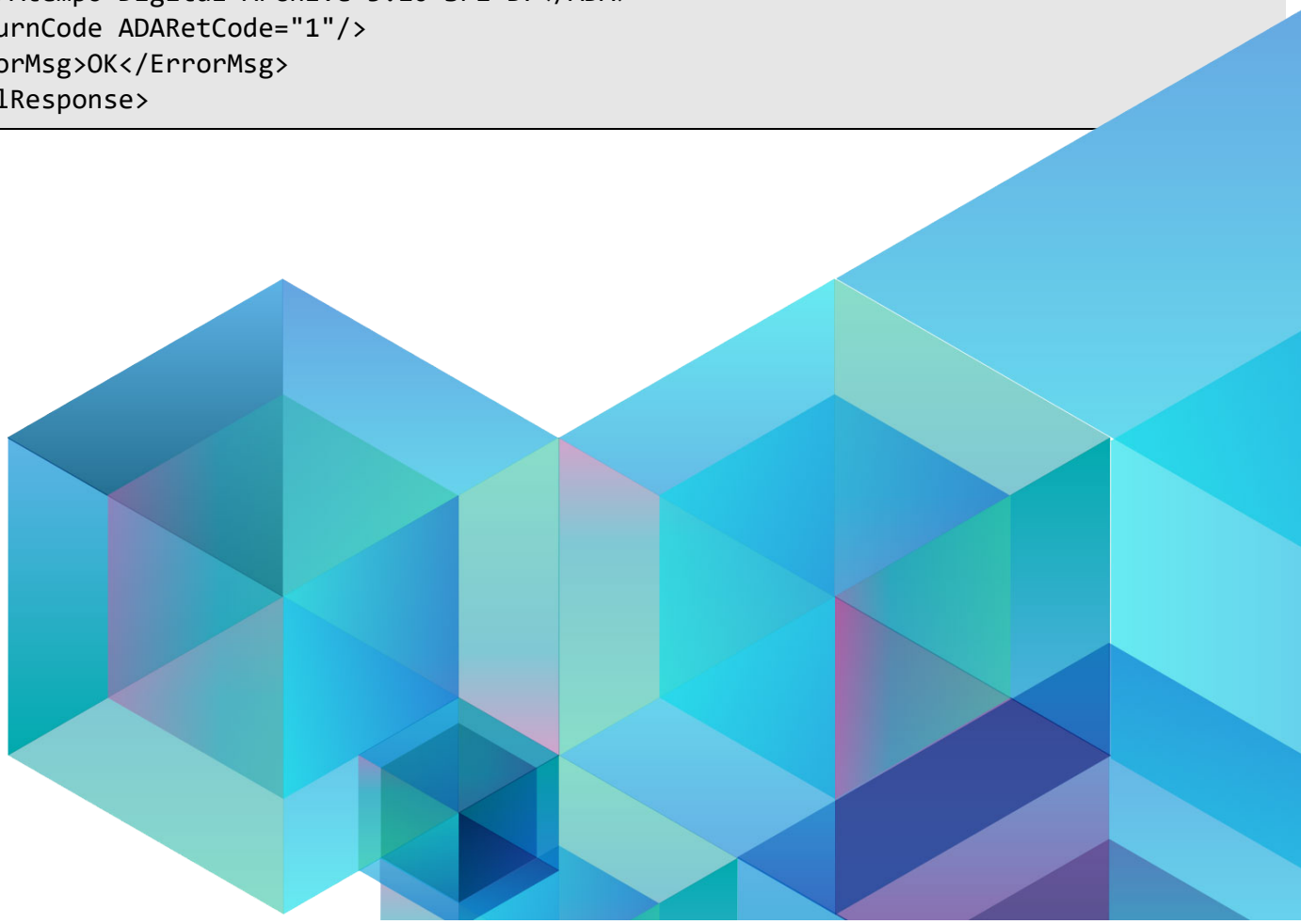
`getServerInfo` returns information about the Miria server installation (server name, HTTP port, HTTPS port).

HTTP Request

```
https://miria-server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b766531/ADA/WS/getServerInfo
```

XML Result

```
<XmlResponse>
<request_date>2020-08-24T15:08:41Z</request_date>
<operation>getServerInfo</operation>
<SERVER_PORT_SSL>443</SERVER_PORT_SSL>
<SERVER_PORT>85</SERVER_PORT>
<SERVER_NAME>adadoc</SERVER_NAME>
<SERVER_REAL_NAME>adadoc</SERVER_REAL_NAME>
<TPE>Tina Perl Engine 2.6 P120 SP0</TPE>
<ADA>Atempo Digital Archive 3.10 SP1 B7</ADA>
<ReturnCode ADARetCode="1"/>
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```



CHAPTER 20 - Retrieving an Archived File

To retrieve an archived file, apply this procedure.

Operation

`retrieveFile` retrieves an archived file.

Options

These are the options that you can use with the `retrieveFile` command:

- `src=`*`archive`*`@`*`path_in_archive`* indicates the path of the archived file or directory that you want to retrieve.
- `dst=[`*`host`*`@]`*`path_on_OS`* indicates the path of the directory or file to which you want to retrieve the archived file. The host specified by `[`*`host`*`@]` must be a Miria agent.

  If the destination is located on the Miria server, it is not required to provide the host name.

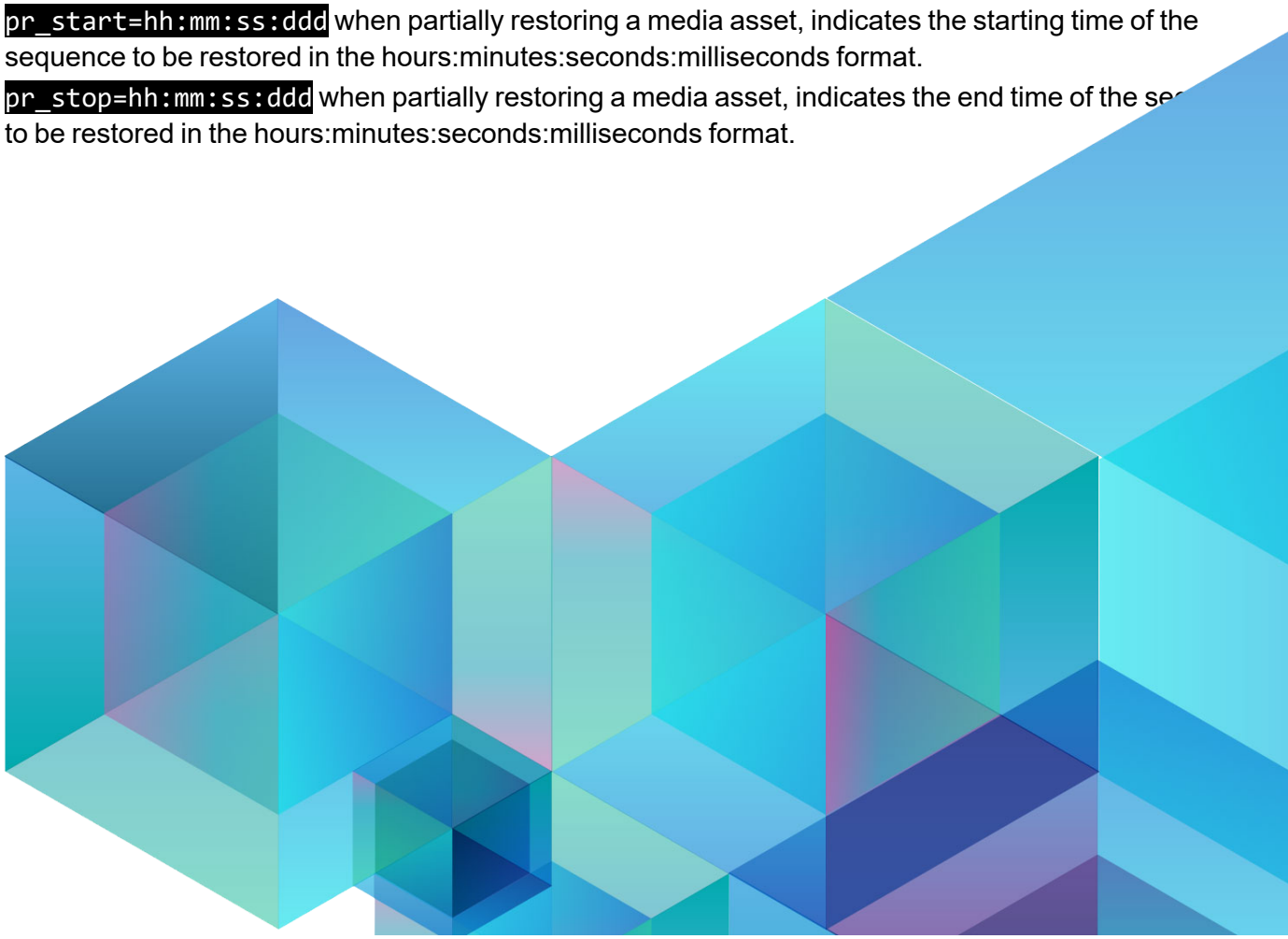
  Renaming the retrieved file.

  For the file to be renamed at retrieval, specify the `dst` parameter in this way:

  ```
  dst=/Dir1/Dir2/file
  ```

  Where: *`Dir1`* and *`Dir2`* are existing directories.
  `file` is the name that you want to give the file after retrieval.

  Example. The `dst=C:\DATA\retrieval\movie1.mxf` command retrieves the `The_Tempest.mxf` file in `C:\DATA\retrieval` and renames it to `movie1.mxf`.
- `pr_start=hh:mm:ss:ddd` when partially restoring a media asset, indicates the starting time of the sequence to be restored in the hours:minutes:seconds:milliseconds format.
- `pr_stop=hh:mm:ss:ddd` when partially restoring a media asset, indicates the end time of the se to be restored in the hours:minutes:seconds:milliseconds format.



- `pr_tc_dest=tcin_zero|tcin_src|same_as_input` when partially restor es what will be the timecode index of the retrieved asset. These values are available:
  - `tcin_zero`: The timecode index for the restored asset starts at `00:00:00:00`, whatever the timecode index in the archived est original asset.
  - `tcin_src`: The timecode index for the restored asset starts at the same value as for the archived original asset.
  - `same_as_input`: The timecode index of the restored asset starts at the timecode index of the archived original asset, plus the input start value.

  Example: If the timecode index of the archived asset starts at `00:24:00:00`, and the clip lasts `45` seconds, and you need to restore starting from second `18`, then the timecode index of the restored asset will start at `00:24:18:00`.
- `debug=1` enables debug traces in the Miria Events.
- `filesList=[host@]path_on_os` indicates the path of a file containing a list of files to retrieve. This option replaces the file option and overrides the previous one. To be valid, this file must contain one filePathInArchive by line. The host specified by `[host@]` must be a Miria agent or a platform name.
- `parallelization_rules=jobs:5 or parallelization_rules=number:100,volume:5 or parallelization_rules=media:1` describes parallelization rules to apply for retrieval tasks splitting. Jobs and media override all other options, but you can combine options number & volume:
  - `jobs:5` is in how many job numbers you want to split your retrieval task.
  - `number:100` is the number of maximum files by job to split your retrieval task.
  - `volume:5` is the maximum volume in GigaBytes by job to split your retrieval task.
  - `media: 1 or 0` is an option to split your retrieval task by media.

Advanced Setting Options

Some options enable you to apply some advanced settings to the retrieval job. These options override the settings defined in the Administration Console.

See Retrieval Settings for details on settings related to retrieval.

- `retrieval_policy_order=policy_name` designates the archiving policy that indicates the order priority of the storage manager containers at retrieval.

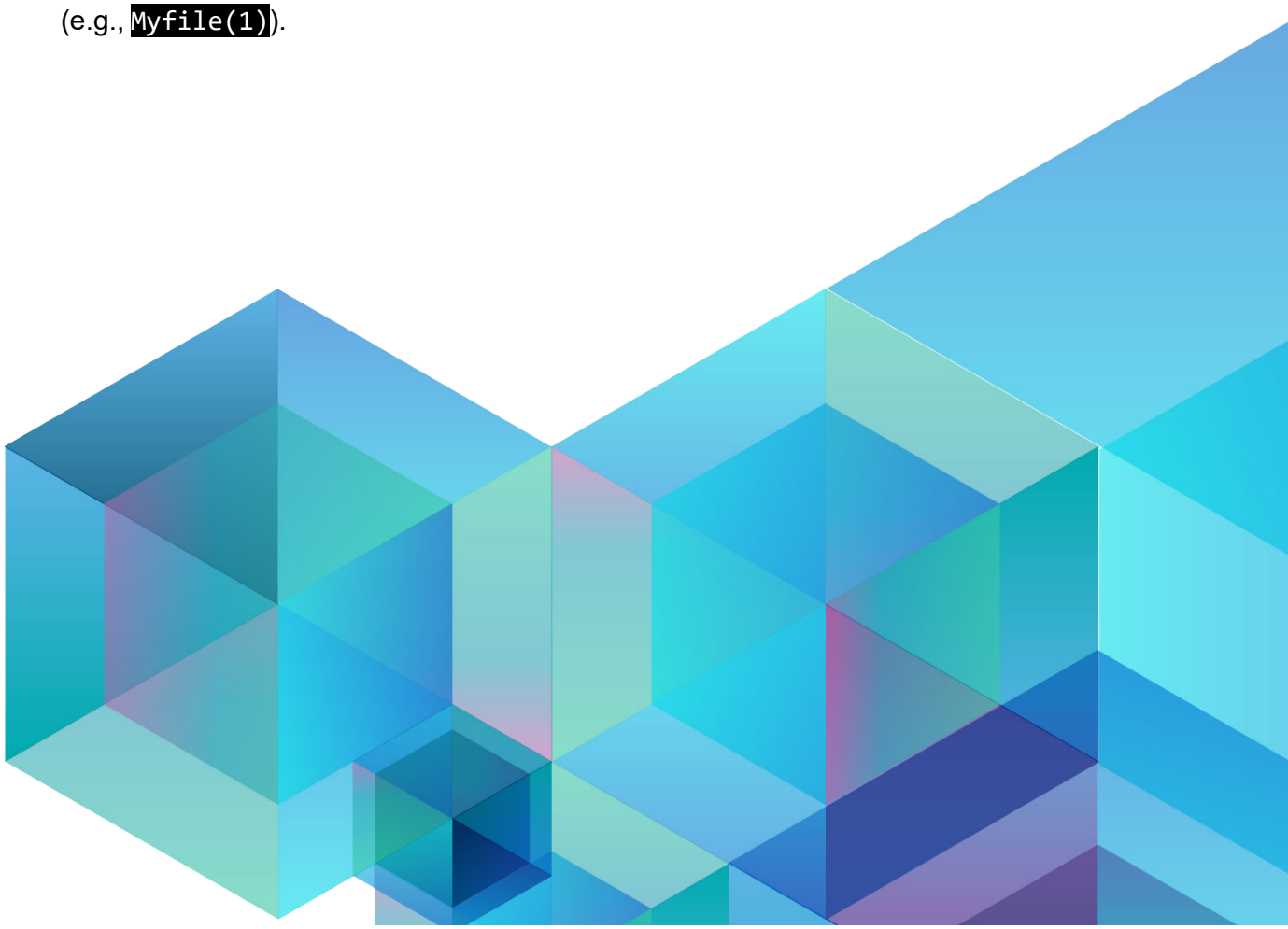
- `parallel_class_retrieval=1|yes|0|no` creates multiple streams for si several files or directories.
- `start_retrieval_media_online=1|yes|0|no` starts the retrieval job only if all media needed for its successful completion are present and available in the library.
- `job_retrieve_altstream=1|0|full` designates the retrieval mode for the alternate streams of archived objects. These are the available options:
  - `1|yes` indicates that Miria retrieves all alternate streams with the actual data (default behavior).
  - `0|no` indicates that Miria does not retrieve alternate streams.
    Use this option when retrieving Windows objects to a UNIX file system, or UNIX objects to a Windows file system as the alternate streams are OS-specific and cannot be interpreted by a different OS.
  - `full` indicates that Miria retrieves files and directories with full access rights granted to all users.
- `fullpath_target=1|yes|0|no` indicates that option `dst=[host@]path_on_OS` is the full path of the retrieved object (i.e., it includes the file name).

  For instance, if the URL includes these options:

  `...src=A@/Myfile&dst=D:\resto\abc\def&create_dir_folder=yes &fullpath_target=yes`, then Miria retrieves the file in `D:\resto\abc` with name `def`.

  However, if the URL includes these options:

  `...src=A@/Myfile&dst=D:\resto\abc\def&create_dir_folder=yes`, then Miria retrieves the file in `D:\resto\abc\def` with name Myfile.
- `create_dir_target=1|yes|0|no` creates the retrieval destination directory automatically if it does not exist.
- `mode=replace|ignore|rename` indicates the behavior that occurs if the retrieved file already exists. These are the available options:
  - `replace` indicates that Miria overwrites the existing file.
  - `ignore` indicates that Miria does not start the retrieval job.
  - `rename` indicates that Miria renames the existing file by adding a number in parentheses to its name (e.g., `Myfile(1)`).



HTTP Request

```
http://frles8loicr/meta/23719251A656889A83BA5C85461E9453/721b796531/ADA/WS/retrieveFile
?src=video@/20160509/WithSuperUser.mov&dst=D:\PERSO\Resto&pr_start=00:00:12:200&pr_
stop=00:00:17:680&pr_tc_dest=same_as_input
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2016-09-02T10:11:17Z</request_date>
<operation>retrieveFile</operation>
<job_id>10512</job_id>
<ReturnCode ADARetCode="1"/>
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

CHAPTER 21 - Retrieving a Folder or an Archived Directory

To retrieve a Folder or and archived directory, apply this Procedure

Operation

`retrieveFolder` retrieves a Folder or an archived directory.

Options

These are the options that you can use with the `retrieveFolder` command:

- `src=`*`archive@path_in_archive`* indicates the path of the folder or archived directory that you want to retrieve.
- `dst=[`*`host@`*`]`*`path_on_OS`* os indicates the path of the directory you want to retrieve the archived directory. The host specified by `[`*`host@`*`]` must be a Miria agent.
- `parallelization_rules=jobs:5 or parallelization_rules=number:100,volume:5 or parallelization_rules=media:1` describes parallelization rules to apply for retrieval tasks splitting. Jobs and media override all other options, but you can combine options number & volume:
  - `jobs:5` is in how many job numbers you want to split your retrieval task.
  - `number:100` is the number of maximum files by job to split your retrieval task.
  - `volume:5` is the maximum volume in GigaBytes by job to split your retrieval task.
  - `media: 1 or 0` is an option to split your retrieval task by media.

Advanced Setting Options

Some options enable you to apply some advanced settings to the retrieval job. These options override the settings defined in the Administration Console.

See Job Settings for details on settings related to retrieval.

- `retrieval_policy_order=p…` …the archiving policy that indicates the order of priority of the storage manager… …retrieval.
- `parallel_class_retrieval=1|yes|…no` creates multiple streams for simultaneous retrieval of several files or directories.
- `start_retrieval_media_online=1|yes|0|no` starts the retrieval job only if all media needed for its successful completion are present and available in the library.
- `job_retrieve_altstream=1|0|full` designates the retrieval mode for the alternate streams of archived objects. These are the available options:
  - `1|yes` indicates that Miria retrieves all alternate streams with the actual data (default behavior).
  - `0|no` indicates that Miria does not retrieve alternate streams.
    Use this option when retrieving Windows objects to a UNIX file system, or UNIX objects to a Windows file system as the alternate streams are OS-specific and cannot be interpreted by a different OS.
  - `full` indicates that Miria retrieves files and directories with full access rights granted to all users.
- `create_dir_target=1|yes|0|no` creates the retrieval destination directory automatically if it does not exist.
- `mode=replace|ignore|rename` indicates the behavior that occurs if the retrieved file already exists. These are the available options:
  - `replace` indicates that Miria overwrites the existing file.
  - `ignore` indicates that Miria does not start the retrieval job.
  - `rename` indicates that Miria renames the existing file by adding a number in parentheses to its name (e.g., `Myfile(1)`).

HTTP Request

```
https://miria-server/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b766531/ADA/WS/retrieveFolder?src=Archive.121@/Folder1/2048_files&dst=E:\retrieve
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2020-06-18T11:47:46Z</request_date>
<operation>retrieveFolder</operation>
<job_id>11</job_id>
<ReturnCode ADARetCode="1"/>
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

CHAPTER 22 - DB Monitoring MaxDB

To monitor the database, apply this procedure.

Operation

`dbMonitor` displays the information on the database monitoring.

Option

This is the option that you can use with the `dbMonitor` command:

`category=general | data | log | backup_history | cache | hotstandby` indicates the information category of the database monitoring to be returned.

If you do not specify any category, then the `dbMonitor` command returns the information on all the categories.

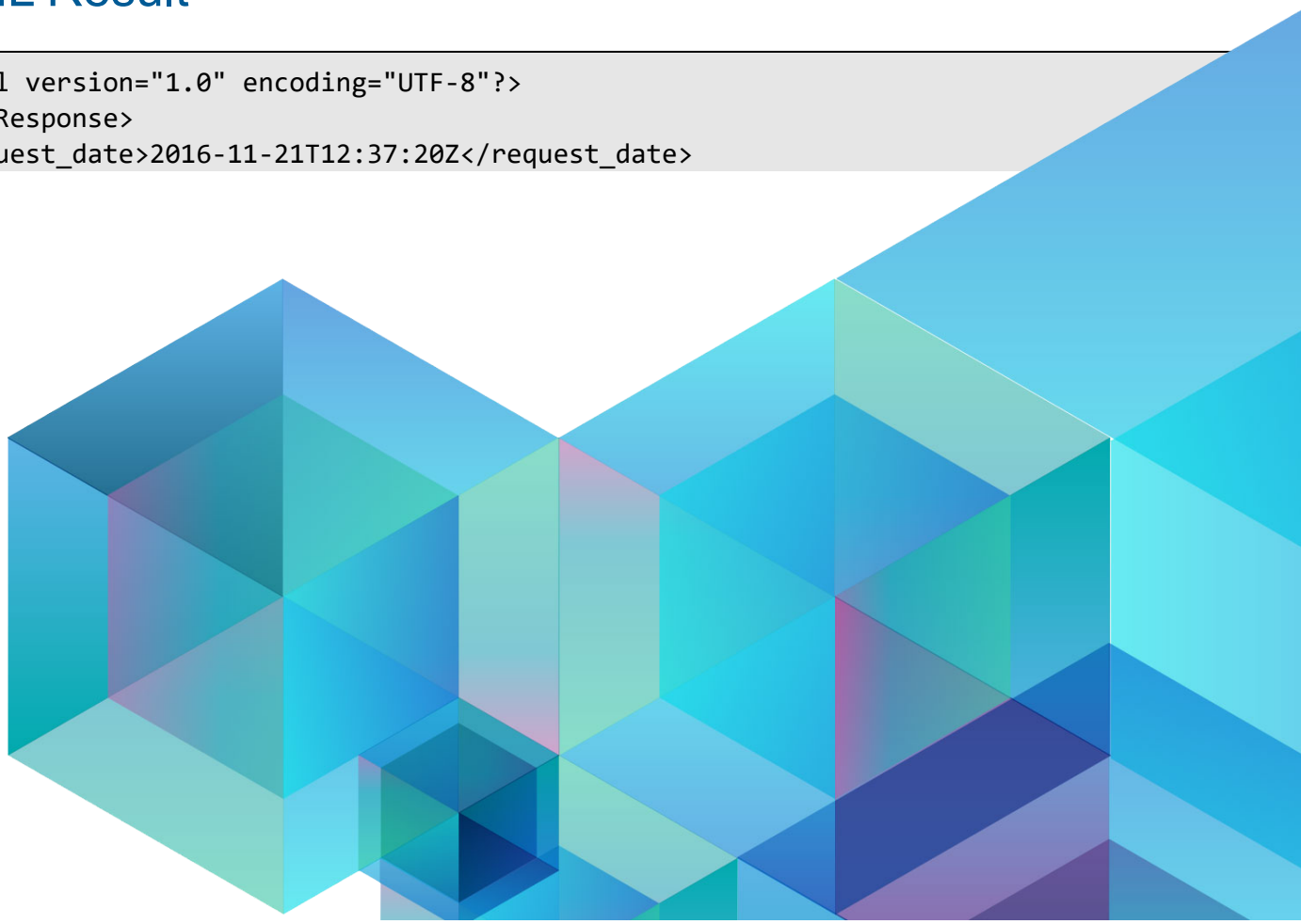
HTTP Request

This HTTP request returns the information on the logs:

```
http://frles8loicr/meta/23719251A656889A83BA5C85461E9453/721b796531/ADA/WS/dbMonitor?category=log
```

XML Result

```
<?xml version="1.0" encoding="UTF-8"?>
<XmlResponse>
<request_date>2016-11-21T12:37:20Z</request_date>
```



```
<operation>dbMonitor</operation>
<ADA>
<LOGFILES>
<FILE_NAME>D:\ADA\Database\MaxDB\Data\ADA\log\DISKL00001</FILE_NAME>
<LOG_PARTITION>1</LOG_PARTITION>
<VOL_NAME>LOG001</VOL_NAME>
<VOL_SIZE_KB>51200</VOL_SIZE_KB>
<VOL_SIZE_PAGES>6400</VOL_SIZ
<VOL_TYPE>F</VOL_TYPE>
</LOGFILES>
</ADA>
<ReturnCode ADARetCode="1"/>
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

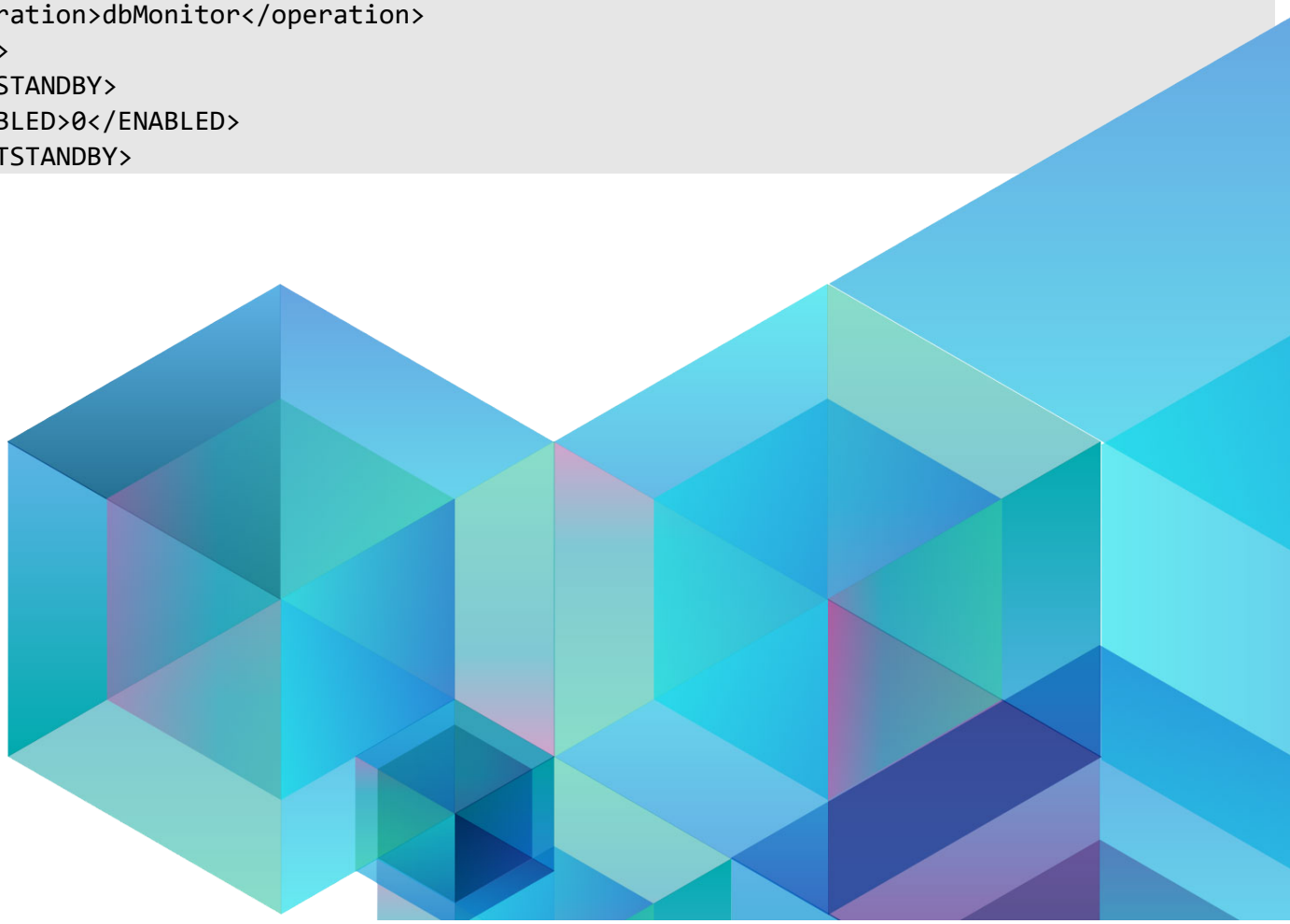
HTTP Request

This HTTP request returns the information on the Hot Standby category:

```
http://frles8loicr/meta/23719251A656889A83BA5C85461E9453/721b796531/ADA/WS/dbMonitor?ca
tegory=hotstandby
```

XML Result

```
<?xml version="1.0" encoding="UTF-8"?>
<XmlResponse>
<request_date>2016-11-21T12:44:01Z</request_date>
<operation>dbMonitor</operation>
<ADA>
<HOTSTANDBY>
<ENABLED>0</ENABLED>
</HOTSTANDBY>
```



```
</ADA>
<ReturnCode ADARetCode="1"/>
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

CHAPTER 23 - DB Monitoring PostgreSQL

To monitor the database, apply this procedure.

Operation

`dbmonitor` displays the information on the database monitoring.

Option

This is the option that you can use with the `dbmonitor` command:

`category=general` indicates the information category of the database monitoring to be returned.

If you do not specify any category, then the `dbMonitor` command returns the information on all the categories.

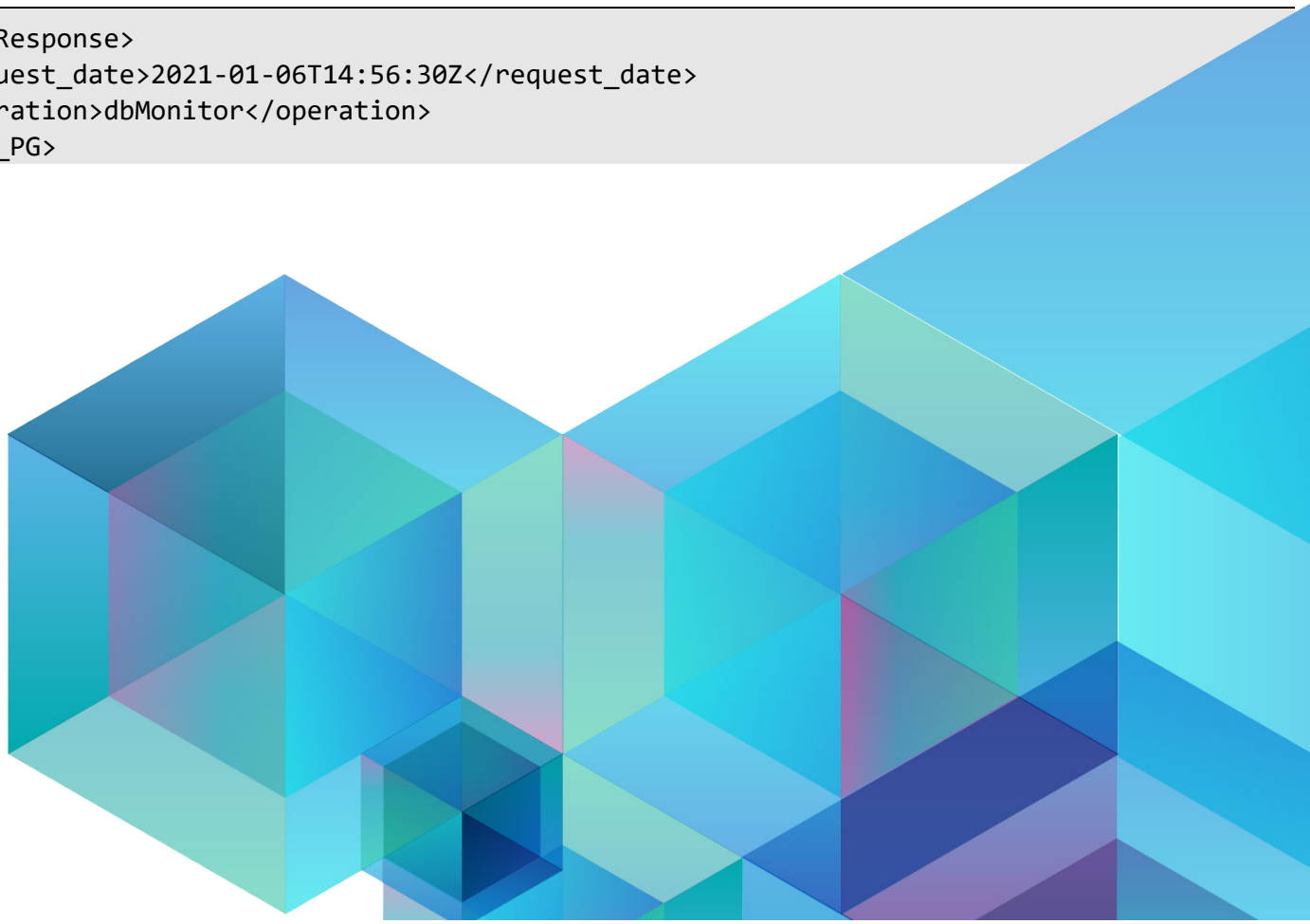
HTTP Request

This HTTP request returns the following information:

```
http://frmsynicod1p/meta/697DB561001E40928EDEFA4961B11A25/721b78660f776501/ADA/WS/dbMonitor?category=general
```

XML Result

```
<XmlResponse>
<request_date>2021-01-06T14:56:30Z</request_date>
<operation>dbMonitor</operation>
<ADA_PG>
```



```
<GENERAL>
<DATABASE_SIZE>15987247</DATABASE_SIZE>
<DB_INSTANCE>ADA_PG</DB_INSTANCE>
<DB_TYPE>POSTGRES</DB_TYPE>
<DB_VERSION>13.0</DB_VERSION>
<NEED_DB_MIGRATE>0</NEED_DB_MIGRATE>
<ONLINE>1</ONLINE>
<STAT_ARCHIVE>
<ARCHIVED_COUNT>0</ARCHIVED_COUNT>
<FAILED_COUNT>0</FAILED_COUNT>
<PG_DATABASE_SIZE>15987247</PG_DATABASE_SIZE>
<STATS_RESET>2020-09-25 10:08:16.228387+02</STATS_RESET>
</STAT_ARCHIVE>
<STAT_REPLICATION>0</STAT_REPLICATION>
</GENERAL>
</ADA_PG>
<ReturnCode ADARetCode="1"/>
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

CHAPTER 24 - Launching a Task

To launch a task, apply this procedure.

Operation

`launchTask` launches the specified task.

Options

These are the options that you can use with the `launchTask` command:

- *`name`* indicates the name of the task.
- *`list`*

  Or

  `execute`

  `list` and `execute` are mutually exclusive.

  In the output of `ada_service -build_url` command, the name `launchTask` is added on the function list.

For an automatic backup task, you must set one of these options:

- `full`. When set to `1` or `yes`, launches a Full automatic backup.

  Or
- `incremental`. When set to `1` or `yes`, launches an Incremental automatic backup.

For an auto-synchronisation task, you could override the source and destination fields in the synchronization task definition by :

- `src`

  Or
- `dst`

> **Example**: launchTask?name=<TaskName>&execute=yes&src=<Sourc

HTTP Request

This HTTP request launches a standard `list` task:

```
https://frles8loicr/meta/813F76D/7218/8653/ADA/WS/launchTask?name=aa_local&list=1
```

This HTTP request launches a standard `execute` task:

```
https://frles8loicr/meta/813F76D/7218/8653/ADA/WS/launchTask?name=aa_local&execute=1
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2017-03-15T14:58:44Z</request_date>
<operation>launchTask</operation>
<job_id>255</job_id>
<ReturnCode ADARetCode="1"/>
<ErrorMsg>OK</ErrorMsg>
</XmlResponse>
```

CHAPTER 25 - Getting Job Events

To get the events of a job, apply this procedure.

Operation

`getJobEvents` displays the events of a job.

Options

These are the options that you can use with the **getJobEvents** command:

- `job_id=id` Identifier of the job for which the events are requested
- `filter=severity1,severity2,severity3,` … Severity filter optional. If no filter provided, all the events are returned, otherwise only events of type provided are returned.

Available severities:

- 0: DEBUG
- 1: INFO
- 2: CRITICAL
- 3: ERROR
- 4: STOP/DIE
- 5: WARNING
- 6: SUCCESS
- 7: STACK
- 8: DEBUG_STACK
- 9: FATAL
- 10: AUDIT_TRAIL

HTTP Request

http://srv:85/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/getJobEvents?job_id=152&filter=3,5,6

XML Result

```
<?xml version=”1.0” encoding=”UTF-8” >
<XmlResponse>
        <request_date>2022-05-18 10:15:43+01</request_date>
        <operation>getJobEvents</operation>
        <job_id>152</job_id>
<event>
        <node>9</node>
        task>2</task>
        <severity>1</severity>
        value>Starting Task Copy(::copy), Job Number 2/2</value>
        <timestamp>2022-02-18 10:15:43+01</timestamp>
</event>
<event>
        <node>12</node>
        <task>…</task>
        <severity>…</severity><value>…</value>
        <timestamp>…</timestamp>
</event>
</XmlResponse>
```

CHAPTER 26 - Duplicate an existing project Archive

To duplicate an existing project archive to a new project archive set, apply this procedure.

Operation

`duplicateArchive` duplicates an existing project archive to a new project archive set. Only its settings are duplicated, not its objects.

Options

These are the options that you can use with the `duplicateArchive` command:

- `archive_src` Name of the source project archive to be duplicated
- `archive_dest` Name of the new project archive. This name must be unique.

Advanced Settings Options

These settings are optional:

- `archive_org_dest` Name of the destination organization folder. Organization folder must already exist. If not provided, duplicated project archive will be created in the same organization folder as the source.

HTTP Request

```
https://frles8loicr/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/duplicateArchive?archive_src=MySourceArchive&archive_dst=MyDuplicatedArchive&archive_org_dest=MyDestinationArchiveOrg
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2011-09-13T15:53:13Z</request_date>
<operation>duplicateArchive</operation>
<ReturnCode ADARetCode="1" />
<ErrorMsg />
</XmlResponse>
```

CHAPTER 27 - Verify Job on a Media

To start a verification job on a media, apply this procedure.

Operation

`duplicateArchive` starts a verification job on a media.

Options

These are the options that you can use with the `verifyMedia` command:

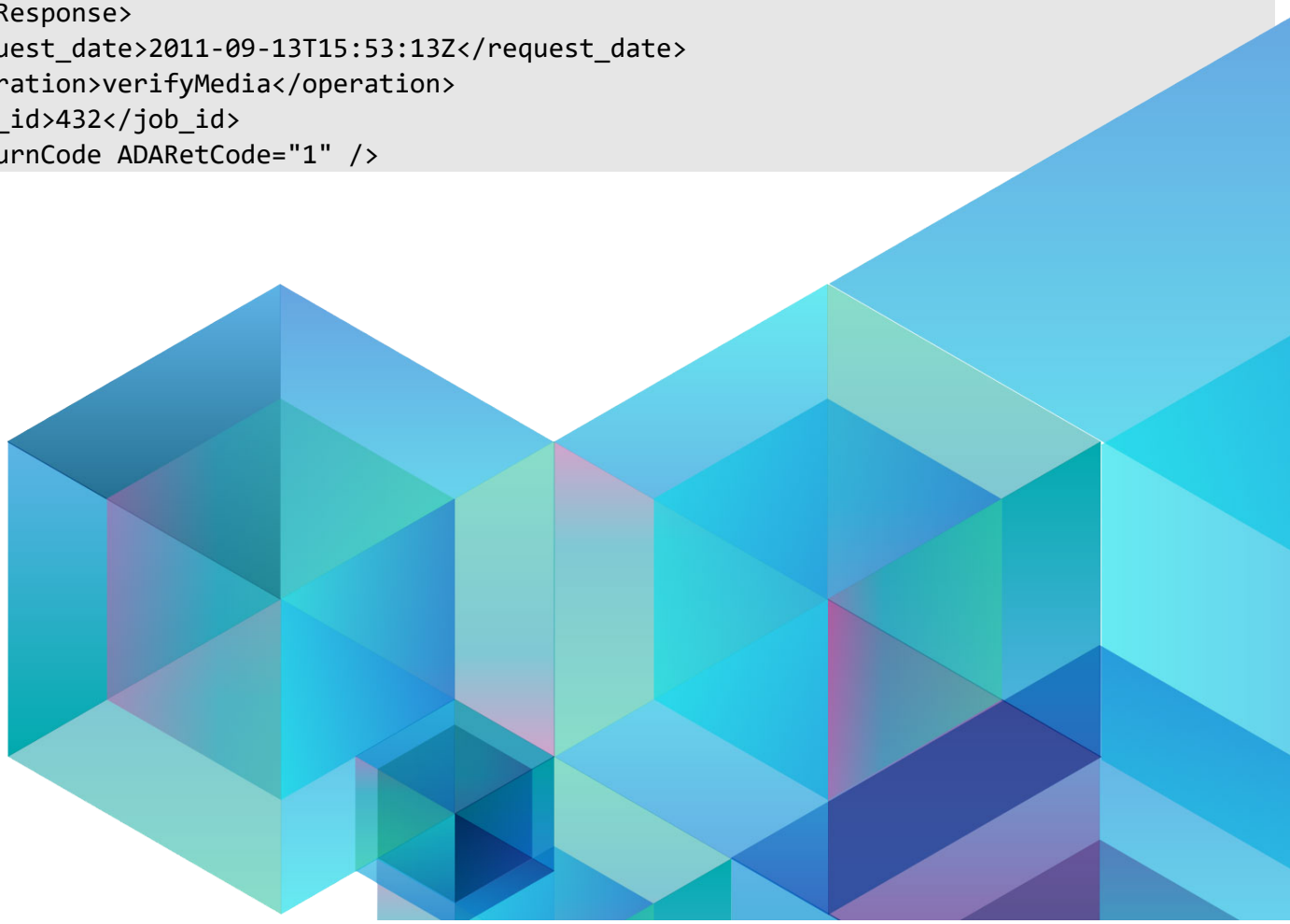
- `barcode` Indicates the barcode of the media to be verified.
- `platform_name` Indicates the name of the platform used to run the verify job.

HTTP Request

```
https://frles8loicr/meta/BD117E7D5CF0912AEA17B4DF33D63F7E/721b736531/ADA/WS/verifyMedia
?barcode=A00001L9&platform_name=MyDataMover
```

XML Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<XmlResponse>
<request_date>2011-09-13T15:53:13Z</request_date>
<operation>verifyMedia</operation>
<job_id>432</job_id>
<ReturnCode ADARetCode="1" />
```



<ErrorMsg />
</XmlResponse>