

XrdOssArc Plug-in Reference

16-July-2025
Andrew Hanushevsky

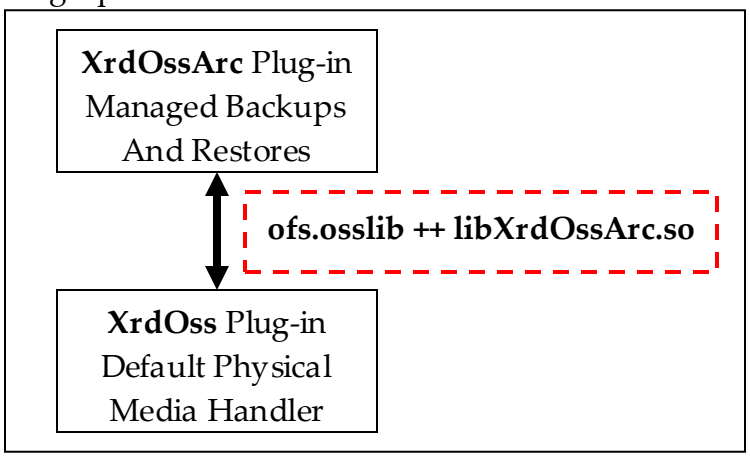


© 2024-2025 by the Board of Trustees of the Leland Stanford, Jr., University; all rights reserved.
Produced by Andrew Hanushevsky for Stanford University under contract
DE-AC02-76-SFO0515 with the Department of Energy.

1 Introduction 5
1.1 Steps in creating a backup 7
1.1.1 Arena Layout 8
1.1.2 Backup Layout 9
1.2 Data Restoration 10
1.2.1 Restore using /archive 11
1.2.2 Restore using /backup 11
1.3 Configuration 12
2 XrdOssArc Directives 15
2.1 arcsize 16
2.2 backup 19
2.3 manifest 21
2.3.1 Manifest File Format 22
2.4 msscmd 23
2.5 paths 25
2.6 rsedcl 27
2.7 rucio 28
2.8 stage 29
2.9 trace 30
2.10 utils 31
3 Script API's 33
3.1 XrdOssArc_Archiver 33
3.2 XrdOssArc_BkpUtils 35
3.3 XrdOssArc_MssCom 37
3.4 XrdOssArc_Weka 38
3.5 Exported Environment Variables 39
4 Document Change History 41

1 Introduction

This document describes the **OssArc** plug-in and its configuration directives. The plug-in implements a backup system for multi-file datasets managed by data management systems (e.g. **Rucio**) by modifying the behavior of a standard **XRootD** data server. The plug-in itself stacks on top of the default Oss plug-in as shown in the graphic below.



The required **oss.osslib** directive specifies the location and name of the shared library plug-in. Here the default linker search path is used (i.e. no directory path is specified). The double plus specifies that the plug-in is to stacked on top of the current **oss** plug-in.

The OssArc plug-in makes use of two virtual paths:

1. **/archive** the root path referring to archive files that are created for backups by the plug-in, and
2. **/backup** the root path referring to files within archives that have been backed up.

The paths do not exist in reality but they need to be visible so that clients can refer to them to retrieve archive files and files within archives. To do this, they must be exported using the **all.export** directive as shown to the right

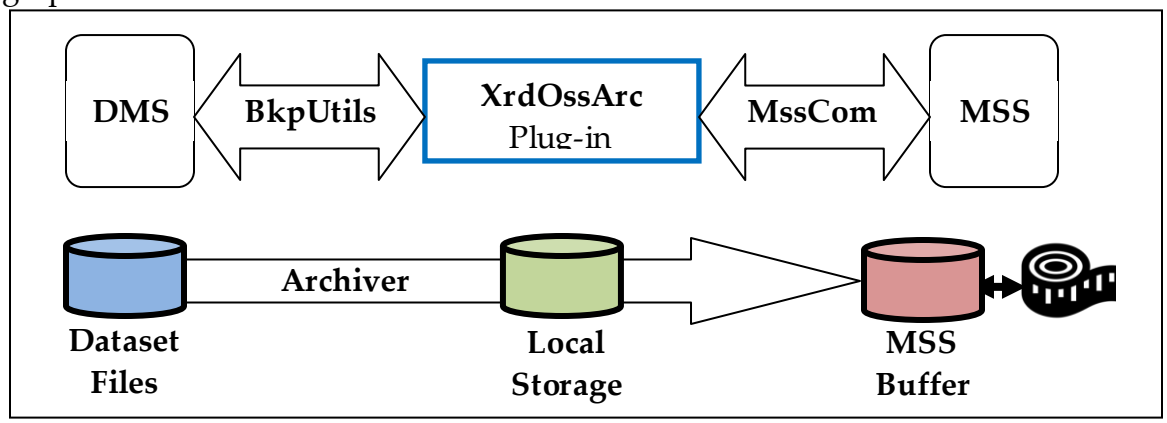
```
all.export /archive r/o
all.export /backup r/o
```

Once the plug-in starts, it interacts with its environment which consists of:

a) The **D**ata **M**anagement **S**ystem (e.g. **Rucio**), **DMS**, to determine which datasets need to be backed up and tell the **DMS** which of them have actually been backed up. It also uses the **DMS** to determine the files in the data sets, their locations, and attributes (e.g. size and checksum),

b) The locally accessible file systems that provide access to the datasets files, the file system to be used to create archive files from the dataset files, the optional fuse mounted file system than is used by the **M**ass **S**torage **S**ystem, **MSS**, as a backup disk buffer (i.e. the buffer that holds the archive files to be transferred to offline media like tape like **HPSS**).

The environment components and interactions are shown in the following graphic.



The **XrdOssArc** plug-in coordinates the data flow to create a backup using three major **Python** scripts:

1. **XrdOssArc_BkpUtils** which communicates with the **DMS** to provide the names of the datasets that need to be backed up, dataset contents, file location, and file metadata.
2. **XrdOssArc_MssCom** which communicates with the **MSS** to provide information on archive location (i.e. buffer or offline media), to bring offline files online (i.e. copy to buffer), and to optionally copy data from local storage to the **MSS** via the network as opposed to a **FUSE** mounted buffer.
3. **XrdOssArc_Archiver** which creates archive files using the file system containing the dataset files and coordinates placing them in the **MSS** buffer either by a direct copy or via an **XrdOssArc_MssCom** network copy.

The plug-in also may make use of two additional **Python** helper scripts:

1. **XrdOssArc_Manifest** which creates a manifest file describing the provenance of the files placed in the archive file. The manifest is also placed in the archive.
2. **XrdOssArc_Weka** which provides optional pre- and post-archive optimizations for **Weka** file systems.

Note that the reference implementation uses **Rucio** as the **DMS** and **HPSS** as the **MSS**. To use a different **DMS**, an appropriate version of **XrdOssArc_BkpUtils** is needed. To use a different **MSS**, an appropriate version of **XrdOssArc_MssCom** is needed.

1.1 Steps in creating a backup

The XrdOssArc plug-in taks the following steps in creating a backup:

1. The plug-in invokes the **XrdOssArc_BkpUtils** script to get a list of datasets that need to be backed up.
    a. The **Rucio** version of the script searchers for datasets that have the metadata key **arcBackup** set to *rsenam*e**:need** where *rsename* is the **RSE** name given the server running the plug-in. Only closed dataset are considered eligible to be backed up.
    b. The script prints the dataset names so that they can be collected by the plug-in.
    c. A
2. The plug-in collects the name of the datasets, removes any duplicates already in progress or queued, and places them in a queue for backup. A queue is necessary because the number of simultaneous backups is configurable.
3. Asynchronously, a set of backup worker threads examine the queue for available work. If a required backup is found, it is removed from the queue and the **XrdOssArc_BkpUtils** script is invoked with the dataset that need to be backed up which performs the following steps:
    a. The script obtains the list of files contained in the dataset. The physical location, size and list of associated checksums is also obtained for each file.
    b. The list is sorted to obtain a deterministic ordering. This is necessary to create an ordinal index.
        i. An ordinal index enables one to trivially determine which of *n* archives comprising a dataset backup contains a desired file.
    c. If a manifest is required, a manifest file is created to be included in the backup Refer to the section describing this file for the format.
    d. The script proceeds to recreate the dataset by creating symlinks to each file in the dataset. This becomes the virtual dataset which is used to create an archive file.
        i. If an archive size limit has been configured, the script creates conforming sets of symlinks. Each set if placed in a separate directory. Each such directory is used to create an archive file of appropriate size.
    e. If a manifest file is to be included in the backup, it is only placed in the first archive file.

4. After the **XrdOssArc_BkpUtils** script sets up the symbolic links representing the archives to be generated in the arena directory, the optional pre-archive script is called.
5. **XrdOssArc_Archiver** script is invoked to create the actual archive files. Each archive file is copied to the **MSS** for eventual creation of an offline backup. The script employs one of two configurable methods:
    a. If the **MSS** disk buffer is **FUSE** mounted on the local host, the archive file is copied to the appropriate location in the **FUSE** mounted file system.
    b. If there is no FUSE mount for the MSS disk buffer or if **XrdOssArc_Archiver** is configured to do a remote copy to the **MSS**, it invokes **XrdOssArc_MssCom** to perform the copy operation as it is specific to the type of **MSS** being used.
6. The optionally configured post-archive script is run.
7. Finally, **XrdOssArc_BkpUtils** script is re0invoked to finalize the backup. The script sets the meta-data **arcBackup** to *rsenam*e**:done** where *rsename* is the **RSE** name given the server running the plug-in. It also removes all artifacts that were created to create the backup (e.g. symlinks and dataset-specific directories specific in the arena directory.

1.1.1 Arena Layout

The configurable arena is a directory path in a locally mounted file system that is used to stage the creation of archive files. The root of the arena is specified by the **oss.localroot** directive (e.g. "**oss.localroot /arc/arena**"). Within that directory, the plug-in creates the path "**dataset/4bkp**". This is where datasets are staged for backup. This directory contains directories corresponding to the scopes that the plug-in must handle. The scopes are configured using the ossarc.backup directives. So, for instance, if datasets in scope prod are to be backed up, you would see

localroot/**dataset/4bkp/prod**

In this directory can be one or more additional directories corresponding to a dataset that needs to be backed up. Dataset names, especially in **Rucio**, typically correspond to a directory path. For instance, "**prod:drp/images/002**" is one such possible identifier. It should be obvious that the scope here is "**prod**" and that becomes a directory but the subsequent path is flattened so that the path appears as

localroot/**dataset/4bkp/prod/drp%images%002**

Flattening directory path is essential to maintaining atomic uniqueness. Since any data set may share a prefix of any other dataset in the same scope, flattening allows the datasets to be uniquely separated from any other dataset in the same scope, facilitating cleanup.

Within the unique dataset path, additional directories are created of the form **"~1**", "**~2**", etc. Each such directory corresponds to an archive file of a subset of files in the dataset. If the datasets is not too large then only a "**~1**" directory exists and all symlinks are placed in that directory. The manifest file, named **Manifest**, is also created at this directory level and included in the first archive file.

This layout makes it trivial to create as many archive files as needed to backup the whole dataset regardless of size. The archiver script merely **cd**'s into each tilde (~) directory and invokes the **zip** command to create an archive file of all of the subsequent contents and place the result in the parent directory. The naming convention is also straightforward which each archive file named as **Archive***m*-*n*.**zip** where *m* corresponds to the numeric part of the tilde directory and *n* is the total number of tilde directories. For instance,

Archive1-3.zip Archive2-3.zip Archive3-3.zip

This also allows one to see exactly how many archive files were created for the backup.

Once the backup finished, cleanup is merely the recursive removal of the unique named dataset directory.

1.1.2 Backup Layout

The backup layout is more traditional than the arena layout. When files can be copied to a **FUSE** mount, the target directory is specified by the **ossarc.paths** directive using the **backing** parameter. For instance, specifying

ossarc.paths backing *fusepath*

causes the archive files to be copied to

fusepath/*scope*/*datasetid*/**Archive***m*-*n*.**zip**

Using the example in the previous section with a single archive file, it would be

fusepath **/prod/drp/images/002/Archive1-1.zip**

When a **FUSE** path is not used, a remote copy program is invoked. The destination root inside the **MSS** is specified using the **ossarc.paths** directive using the **mssfs** parameter. For instance, specifying

ossarc.paths mssfs *msspath*

The layout is still identical except that instead of using the *fusepath* the archiver uses *msspath* via a remote copy program as the root path.

1.2 Data Restoration

The **XrdOssArc** plug-in allows two types of data restoration:

1. full archive restore and
2. by individual file.

The type of restore requested is determined by the path used to reference a dataset or archive. Specifically,

a) Fetching data by the root path **/archive** is taken to be a reference to a complete archive file.
b) Fetching data by the root path **/backup** is taken to be a reference to an individual file for restoration.

In order for this to work, the **XRootD** server needs to minimally export **/archive** and **/backup** to allow clients to fetch data for restoration. Furthermore, currently restoration depends on the presence of a **FUSE** mount to access the **MSS** to fetch the archives needed to perform a restore. Future version will allow restores using a remote copy command similar to what is done for backups.

In any case, the working model for restores is that a client simply copies the requited data (file or archive) to the location that the restoration needs to happen. This can be done using **xrdcp** or, should **HTTP** be configured, using an **HTTP** copy program (e.g. **curl** or **wget**).

Whether using **/archive** or **/backup**, the path always names a dataset as this is what is being restored either in part or whole; that is, /**archive/***dataseetid* and **/backup/***datasetid* where *datasetid* is the name of the dataset in the form of *scope***:***datasetpath*. Using the previous example, "**prod:drp/images/002**" would be the full *datasetid*.

The following sections detail how **/archive** and **/backup** requests are handled.

1.2.1 Restore using /archive

In the case of restoring data using archive files, the idea is to copy out the archives that were created as a backup for the dataset. Since a backup may consist of one or more archive files. To list the archive files comprising a backup use '**xrdfs ls**' or an **HTTP HEAD** request using **/archive/***datasetid* as the path. Once you have the archive list, proceed to copy each one using the **CGI** key **ossarc.fn** to specify the archive filename. For instance, **if /archive/***datasetid* has two archive files:

Archive1-2.zip and **Archive2-2.zip**

you would copy out
/archive/*datasetid***?oss.fn=Archive1-2.zip** and
/archive/*datasetid***?oss.fn=Archive2-2.zip**

Currently, recursive copies are not supported.

Alternatively, if you only want an archive file that contains a specific file, you can simply copy out the archive that contains the file specifying

/archive/*datasetid***?ossarc.fn=***datasetfile*

Using "**prod:drp/images/002**" as the *datasetid* and the file of interest within that dataset being "**prod: :dir/dir_a/myfile**" you simply copy out

/archive/prod:drp/images/002?ossarc.fn=prod: :dir/dir_a/myfile

Be aware that full archive containing that file is copied out not the individual file.

You can also list the name of the archive file that contains a particular dataset file using **ls** or **HEAD** with the same *path*?*cgi* specification.

1.2.2 Restore using /backup

Restoring individual files is similar to copying out an archive file except that the root path is /backup, as shown below

/backup/*datasetid***?ossarc.fn=***datasetfile*

Where: *datasetfile* is the name of the file in the dataset named *datsetid*. Using "**prod:drp/images/002**" as the *datasetid* and the file of interest within that dataset being "**prod: :dir/dir_a/myfile**" you simply copy out

/backup/prod:drp/images/002?ossarc.fn=prod: :dir/dir_a/myfile

The specification is used in the source **URL** for **xrdcp** or any **HTTP** fetch program (e.g. **curl** or **wget**). Be aware that only that file is copied out after the archive holding the file is retrieved from the **MSS** which may be somewhat lengthy.

1.3 Configuration

Configuring the **XrdOssArc** plug-in simply means adding specific configuration directives in the usual **xrootd** configuration file. This section covers only things that are specific to the plug-in. Refer to **Xrd/XRootD Configuration Reference** on how to configure the basic **xrootd** server, and enable **HTTP** access.

Below are the required directives to get the plug-in to load and configure itself for a **Rucio DMS** and a **HPSS MSS**.

```
# Set the environmental variables that the Rucio client needs
# to communicate with the Rucio server.
#
setenv RUCIO_CONFIG = config_file_path
setenv RUCIO_ACCOUNT = username

# Identify the names of the two participating parties
#
ossarc.rsedcl data_source_rse_name backup_rse_name

# Specify the command to use when communicating with the MSS.
# For HPSS, the hsi command is used (see subsequent text).
#
ossarc.msscmd command_and_arguments

# Specify where the MSS disk buffer is FUSE mounted and the
# root path inside the MSS where archives are placed.
#
ossarc.paths backing path_to_MSS_FUSE_mount
ossarc.paths mssfs path_inside_the_MSS_holding_archive_files

# Make sure that the virtual archive and backup paths are
# visible and useable to the outside world.
#
all.export /archive r/o
all.export /backup r/o

# Set the local root for the arena, the local file system to
# be used for creating archive files to be backed up.
#
oss.localroot local_root_path

# Load the plug-in
#
ofs.osslib ++ libXrdOssArc.so
```

While the **XrdOssArc_MssCom** script that communicates with the **MSS** used for backups is relatively general, it does assume certain **HPSS**-like interfaces. One of them is a single command that can be used not only to communicate with the **MSS** but to also copy files in and out of the **MSS**. The **ossarc.msscmd** directive specifies that command. For **HPSS** it is **hsi**. The command requires several options and is usually packaged as a script and the path to the script is actually specified as the argument to the **ossarc.msscmd**. Below is the typical script that is used.

```
/usr/local/bin/hsi -q -A keytab -k path_to_keytab -l loginid \
$@
```

While the above could have been specified in the configuration file, specifying an intermediate script allows for changes without needing to restart the server.

Also notice that the **ossarc.rsedcl** command is somewhat **Rucio** specific in that the names specified should be the names configured in **Rucio** as **R**emote **S**torage **E**lements (**RSE**). This also specifies who is responsible for what. The first name identifies the **RSE** that is being backed up. The second name identifies the **RSE** that is doing the backup. While this is not strictly a one-to-one relationship, it does restrict a single **xrootd** server to backing up a single **RSE** as only one **ossarc.rsedcl** directive may be specified. Typically, if an **RSE** is named *xyzzy* the corresponding **RSE** doing the backups should be named *xyzzy*_**BACKUP** to avoid any confusion. This is a logical constraint and is not enforced. It is possible to have one backup **RSE** serving multiple data source **RSE**'s. However, you will need to run an instance of **xrootd** for each data source **RSE** with a configuration file specific to the data source.

There are additional **XrdOssArc** directives that you can specify to customize the back and restore operation.

2 XrdOssArc Directives

The following list links to directives by general function.

- Backup execution parameters and scope
  - arcsize - archive file limits
  - backup - scheduling and mode
  - manifest - manifest file contents
- Debugging
  - trace - verbiage level
- Data Management System Identification
  - rsedcl - names of Remote Storage Elements (RSE)
- Environmental Information
  - msscmd - command to interact with the MSS
  - paths - location of storage areas and scripts
  - utils - the scripts to be used
- Restore Execution and Scope
  - stage - MSS staging limits
- Rucio Specific Parameters
  - rucio - Rucio limits

2.1 arcsize

```
ossarc.arcsize optsz [range minsz maxsz] [skip]
```

Function

Specify the desired size of an archive file.

Parameters

optsz The optimal size for an archive file. The *optsz* may be suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabyte, or **g**igabytes, respectively. The default is bytes.

minsz The minimum acceptable size for an archive file. The *minsz* may be suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabyte, or **g**igabytes, respectively. The default is bytes. Specify a size that is less than or equal to *optsz*. See the notes for the default when **range** is not specified.

maxsz The maximum acceptable size for an archive file. The *maxsz* may be suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabyte, or **g**igabytes, respectively. The default is bytes. Specify a size that is greater than or equal to *optsz*. See the notes for the default when **range** is not specified.

skip Skips archiving the dataset when it is the **range** restriction cannot be honored. The dataset is marked as "**skip**" indicating that it was not backed up. See the notes on the action taken when **skip** is not specified (i.e. the default action).

Defaults

See notes.

Notes

1) When **arcsize** is not specified, no size restrictions are applied. A single archive file is created named 'Archive1-1.zip'.
2) When range is not specified, the *minsz* defaults to *optsz*/2 and the *maxsz* defaults to *optsz**2.
3) When **skip** is not specified, the system may violate the **range** restriction in order to produce an archive. It attempts to come as close as possible to the restrictions while favoring larger archive files.

Example

```
ossarc.arcsize 1g range 1g 2g
```

2.2 backup

```
ossarc.backup [fscan fsec] [max num]
              [minfree mnf[%|k|m|g]]
              [mode {local | remote}]
              [stopchk ssec] [poll psec]
              scope sname [ sname [. . .]]
```

Function

Specify backup activity parameters.

Parameters

fscan *fsec*

Specifies how often the **FUSE** mounted backup file system should be queried for usage statistics. This information is used to determine whether or not backup activity must be suspended until enough free space is recovered. The default is *psec*/3 but no less than 30. Specify a value greater than or equal to 30. The *fsec* can be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the default).

max *num*

Specifies the maximum number of backup tasks that may be run in parallel. The default is 2.

minfree *mnf*

Specifies the minimum amour of free space than must exists in the **FUSE** mounted file system used to backup archive files to the Mass Storage System. When the amount falls below this threshold, backups are suspended until sufficient free space becomes available. When *mnf* is suffixed with percent (**%**) the *mnf* value is calculates as a percentage of the allocated space. Otherwise, the value is taken as a fixed number of bytes. The *mnf* may be suffixed by **k**, **m**, or **g** to indicate **k**ilobytes, **m**egabyte, or **g**igabytes, respectively; otherwise, it is assume to be bytes. The default is 20%.

mode Specifies how data is transmitted to the Mass Storage System used to backup data. Specify one of two options:
local - use the local **FUSE** mount, the default.
remote - use a remote transfer agent (see notes).

poll *psec*
Specifies the frequency at which **Rucio** will be scanned for datasets that need to be backed up. Specify a value greater than or equal to 60 seconds. The *psec* can be suffixed by **h**, **m**, or **s** to indicate **h**ours, **m**inutes, or **s**econds (the standard). The default is 15m.

scope *sname* […]
Specifies one or more **Rucio** scopes where datasets eligible for backup reside. At least one scope must be specified as there is no default. The scope specification must be the last specification for the directive.

Example
```
ossarc.backup mode remote scope prod
```

Notes
1) When mode is selected to be remote, the **XrdOssArc_Archiver** script calld the **XrdOssArc_MssCom** script to copy the archive files to the Mass Storage System using the configured transfer program. Otherwise, the files are copied to the **FUSE** mounted file system for backup by the Mass Storage System.
2) The calling sequence in remote mode is:
XrdOssArc_MssCom save *mss_dir_path* **./***fname* [**./***fname* […]]
Where *mss_disr_path* is the absolute path in the target storage system where the subsequently specified files (i.e. **./***fname*I) are to be placed.
3) The command that is used to transfer files via the **XrdOssArc_MssCom** script is specified using the **ossarc.msscmd** directive.

2.3 manifest

```
ossarc.manifest parms

parms:   cksum {csname | none}
```

Function

Specify manifest parameters.

Parameters

parms Are one or more parameters specific to the directive. At least one parameter must be specified.

cksum

Specifies which checksum is to be included in the manifest. Specify a valid **Rucio** checksum name for *csname*. If *csname* is specified as **none** then no checksums are included in the manifest. The default *csname* is **adler32**.

Defaults

assarc.manifest cksum adler32

Notes

1) When the specified *csname* is not available via a **Rucio**, the manifest substitutes "**None**" for its value.
2) The manifest is a file that describes the provenance of the data residing in the archives. It is always included in the first archive file. Refer to the following section on the manifest file format.

2.3.1 Manifest File Format

The manifest file whose name is "**Manifest**" in the archive, is a literal **Python** data structure that can be converted to a run-time copy using the **ast** library's **ast.literal_eval()** function The structure is a list of tuples laid out as:

```
[[<lfn>, <pfn>, <bytes>, {<cksname>: <cksval> ...}] ...] # Source-RSE=rse_name
```

Where:

lfn is the logical file name.

pfn is the physical file name that was used to make a copy of the file.

bytes the size of the file in bytes.

csname
: is the dictionary key for the name of checksum name of the recorded checksum value. This is specified using the **ossarc.manifest** directive.

csnval is the corresponding checksum value recorded as a hexadecimal character string. The value is obtained from **Rucio**.

rse_name
: is the **Rucio** name of the **RSE** that provided the data.

Each element in the list corresponds to a file in the archive and is a tuple of the form **[str, str, int, dict**]. The list is in the order that files were written into the archive.

2.4 msscmd

```
ossarc.msscmd cmd
```

Function

Specify the command to be used to communicate with the Mass Storage System used to backup archives.

Parameters

cmd Is the actual command to be used by the **XrdOssArc_MssCom** script to perform functions relative to the **MSS** being used.

Defaults

There is no specific default. The **XrdOssArc_MssCom** script is responsible for providing any defaults.

Notes

1) The specified *cmd* is exported in the **XRDOSSARC_MSSCMD** environmental variable.

Example

```
ossarc.msscmd /usr/local/bin/hsi -q -A keytab -k /etc/his/keytab -l hsiadm
```

2.5 paths

```
ossarc.paths parms

parms: [backing bpath] [mssroot mpath] [srcdata spath]

       [stopfile tpath] [utils upath]
```

Function

Specify the file system locations for various information needed to perform backups and restores.

Parameters

backing *bpath*

Is the path in the **FUSE** mount point to store archive files into or fetch from the Mass Storage System. The default is **"/TapeBuffer**".

mssroot *mpath*

Is the path inside the Mass Storage that corresponds to the **backing** *bpath* when communicating with the Mass Storage System using the **XrdOssArc_MssCom** script. There is no default. This path *must* be specified.

srcdata *spath*

Is the path to the file system that holds all of the data eligible for archiving. There is no default. This path *must* be specified.

stopfile *tpath*

Is the path where the **STOP** file will appear to drain server processing. The default is the server's **adminpath**.

utils *upath*

Is the path where the XrdOssArc Python script utilities reside. The default is "/usr/local/etc".

Defaults

ossarc.paths backing /TapeBuffer utils /usr/local/etc

Notes

1) At least one parameter must be specified.
2) Other than the **utils** path, all other paths should be specified. This assumes that the default **utils** path hosts the **XrdOssArc_xxx** scripts.
3) If you have configured remote backup mode and are not allowing restores, then the backing path is essentially meaningless. However, it must still exist and be writable even if it will not be used.

2.6 rsedcl

required

ossarc.rsedcl *srcrse thisrse*

Function

Identify the participating RSE's.

Parameters

srcrse Is the **Rucio RSE** name that is supplying the data being backed up. This is also the **RSE** hosting the data located at the path specified by **ossarc.paths srcdata** directive.

thisrse Is the **Rucio RSE** name of the server executing the backs ups (i.e. this **RSE**).

Defaults

None, this directive must be specified.

Notes

None.

2.7 rucio

```
ossarc.rucio parms

parms:   [maxitems num]
```

Function

Specify **Rucio** specific constraints.

Parameters

maxitems *num*

Is the maximum number of items that may be queried and returned by the **Rucio Python API**. Specify for *num* the same value specified for the corresponding setting in the **Rucio schema.**py file. The default is 1000.

Defaults

```
ossarc.rucio maxitems 1000
```

Notes

1) Failure to specify the same value for **maxitems** that is specified in the **schem.py** file leads to failure when the value is greater than one in in schema.py) or inefficiency when it is smaller than the one in schema.py.

Example

```
ossarc.rucio maxitems 5000
```

2.8 stage

```
ossarc.stage  parms

parms:  [max num] [poll sec]
```

Function

Specify the staging parameters for data recovery operations.

Parameters

max *num*

Is the maximum number of staging operations allowed at one time. Specify for *num* a value from 1 to 100, inclusive. The default is 10.

poll *sec*

Is the frequency, in seconds, at which clients are asked to revisit when data is awaiting transfer to disk (i.e. poll for readiness). Specify for sec a value from 5 to 100, inclusive. The default is 30.

Defaults

```
ossarc.stage max 10 poll 30
```

Notes

1) At least one parameter must be specified.

Example

```
ossarc.stage max 3 poll 15
```

2.9 trace

```
ossarc.trace parms

parms: {debug | off | save} [parms]
```

Function

Specify debugging parameters.

Parameters

debug

prints detailed information about the execution flow.

off turns off all current debug settings. This is the initial default.

save disables cleanup after a backup operation.

Defaults

ossarc.trace off

Notes

1) The **save** setting allows you to see how files were laid out for archiving.
2) Archiving layout starts at '*oss.localroot*/**dataset/4bkp/**'. The **oss.localroot** directive is used to set the root path in the file system.
3) Tracing incurs substantial overhead and should only be used to help solve problems.

Example

ossarc.trace debug

2.10 utils

```
ossarc.utils parms

parms:   {archiver apath | bkputils bpath | msscom mpath |
          postarc popath | preparc prpath | saver spath}
          [ parms ]
```

Function

Specify the location and names of utility scripts.

Parameters

archiver *apath*

The script that produces archive files and moves them to the Mass Storage System. Specify for *apath* the name of the script optionally prefixed with a path. The default is **XrdOssArc_Archiver**.

bkputils *bpath*

The script that communicates with Rucio. This script drives all data management functions, provides information on what needs to be backed up, and sets up the dataset to be backed up by the **archiver**. Specify for *bpath* the name of the script optionally prefixed with a path. The default is **XrdOssArc_BkpUtils**.

msscom *mpath*

Specifies the script that communicates with the Mass Storage System. This script drives all MSS related functions, provides information on what backups are available, and assists the restoration process. Specify for m*bpath* the name of the script optionally prefixed with a path. The default is **XrdOssArc_MssCom**.

postarc *popath*

The script that should be called after the **archiver** script completes but before it cleans up the staging area. Specify for *popath* the name of the script optionally prefixed with a path. There is no default and if the parameter is not specified, no script is called.

prearc *popath*

The script that should be called before the **archiver** script is called. Specify for *prpath* the name of the script optionally prefixed with a path. There is no default and if the parameter is not specified, no script is called.

saver *spath*

The script that can copy archive files to the Mass Storage System. This scipt is called when backup mode is remote (see the ossarc.backup directive). Specify for *spath* the name of the script optionally prefixed with a path. The default is whatever **msscom** *mpath* is set to.

Defaults

```
ossarc.utils archiver XrdOssArc_Archiver \
             bkputils XrdOssArc_BkpUtils \
             msscom XrdOssArc_MssCom
```

Notes

1) The actual location of the specified script depends on whether the specification starts with a slash or not. When it starts with a slash, it is taken as an absolute path and used as-is. Otherwise, it is considered a relative path and the specified or default **ossarc.paths** directive **utils** path prefixes the specification or default value.

Example

```
ossarc.utils msscom /usr/local/etc/mymsscom prearc mysetup
```

3 Script API's

3.1 XrdOssArc_Archiver

```
XrdOssArc_Archiver dsndir mssdir arcfnt [copycmd]
```

Function

Create one or more archive files and save them.

Arguments

dsndir the directory where archives are to be placed. This directory also contains the ~*n* directories each containing the data to be used to create an archive file.

mssdir the directory where the archive files should be copied to, this either corresponds to the **FUSE** mounted **MSS** disk buffer when no *copycmd* is supplied or the destination directory in the **MSS** that the *copycmd* is to use.

arcfnt the archive filename template (e.g. Archive.zip).

copycmd

the command to use to copy the archive files to the **MSS**. If not supplied, the archive files are simply copied to the *mssdir*.

Notes

1) The *copycmd* is executed as "*copycmd* **save** *mssdir arcfiles*" where *arcfiles* is the list of archives to be copied to *mssdir*.

3.2 XrdOssArc_BkpUtils

```
XrdOssArc_BkpUtils args

args:    addkey key [key […]]

         finish  rse scope dsname dsndir finkey finval

         list    key value scope eolval

         qkey    key[,key[,…]] scope dsname

         set     key value scope dsname

         setup   rse scope dsname areadir mntpfn [manpfn]

         stat    cgi scope did

         which   arcfn fname scope dsname
```

Function

Perform function requiring interaction with the data management system.

Arguments

Notes

1)

3.3 XrdOssArc_MssCom

```
XrdOssArc_MssCom args

args:    evict path [path […]]

         offline path

         online  path

         save    mssdir ./file [./file […]]
```

Function

Perform functions requiring communications with the Mass Storage System.

Arguments

Notes

1)

3.4 XrdOssArc_Weka

```
XrdOssArc_ Weka {dispose | prepare} scope manpath
```

Function

Perform functions requiring communications with the **Weka** file system.

Arguments

dispose

Archive files have been saved, any tier-1 storage reserved for the source files contained in the archives may be relinquished.

prepare

Archive files are to be created, the source files contained in the archives should be placed in tier-1 storage.

scope the **Rucio** scope for this archive operation.

manpath

The paths to the manifest file which contains the list of files that the archives will contain.

Notes

1)

3.5 Exported Environment Variables

The following table shows the environment variable exported by **frm_purged**. These may be used by external programs and plug-ins, as needed. They should never be modified.

Variable	Contents
XRDOSSARC_CKSUM	The name if the checksum value to be included in the manifest file for each dataset file. If not set, checksums are not included in the manifest.
XRDOSSARC_DEBUG	When set contains the debug level: 1 for low and 2 for high.
XRDOSSARC_DSTRSE	Contains the name of this RSE.
XRDOSSARC_MAXITEMS	The maximum number of items that may be fetched from Rucio. If not set, 1000 is assumed.
XRDOSSARC_MSSCMD	The command to use when communicating to the MSS.
XRDOSSARC_MSSROOT	The root directory path to use for files in the mss when using the MSS command.
XRDOSSARC_SIZE	The constraints on the archive file size. The value is a list of three numbers: desired size, minimum size, maximum size, and a Boolean flag. When the flag is true, the archiving to fail when the parameters cannot he honored. When not set a single archive file is created.
XRDOSSARC_SRCRSE	Contains the name of the RSE being backed up.

4 Document Change History

16 Jul 2025

- New Document.