

---

# **ecs-sync Documentation**

***Release 3.1***

**DelIEMC**

**Jun 22, 2017**



---

## Contents

---

<b>1</b>	<b>General Performance Metrics</b>	<b>1</b>
1.1	General Performance ecs-sync 3.0 OVA With Validation . . . . .	1
1.2	Real World Examples . . . . .	1
<b>2</b>	<b>Installing the Service</b>	<b>3</b>
2.1	Updates and vulnerabilities . . . . .	4
<b>3</b>	<b>Creating Config File</b>	<b>5</b>
3.1	Please note: . . . . .	5
3.2	The new generator is part of the ecs-sync-ctl tool and invoked like so: . . . . .	5
<b>4</b>	<b>Plugin Support</b>	<b>9</b>
4.1	Available plugins are listed below along with any custom options they may have . . . . .	9
<b>5</b>	<b>Initializing the UI</b>	<b>21</b>
5.1	Quick Start . . . . .	21
<b>6</b>	<b>Getting Started</b>	<b>23</b>
6.1	Installing the UI . . . . .	23
6.2	Logging In . . . . .	23
6.3	Starting a sync . . . . .	24
<b>7</b>	<b>CLI Syntax</b>	<b>25</b>
<b>8</b>	<b>Starting the ecs-sync Service</b>	<b>39</b>
8.1	Preparing the Configuration File . . . . .	39
8.2	Starting a Sync . . . . .	39
8.3	Checking status . . . . .	40
8.4	Changing Thread Count . . . . .	40
8.5	Pausing/Resuming/Stopping . . . . .	40
8.6	Deleting a job . . . . .	41
<b>9</b>	<b>Troubleshooting the UI</b>	<b>43</b>
9.1	Grails Error on First Login . . . . .	43
9.2	Attempting to Initialize Shows “Missing Configuration” Error . . . . .	43
9.3	“Sync Options” fields required when creating a new sync . . . . .	43
9.4	Unexplained CAS object failures . . . . .	43

<b>10</b>	<b>What is ecs-sync?</b>	<b>45</b>
<b>11</b>	<b>Why use ecs-sync?</b>	<b>47</b>
<b>12</b>	<b>What it Does</b>	<b>49</b>

# CHAPTER 1

---

## General Performance Metrics

---

It is important to note that the performance metrics posted below are real world examples but are meant to serve only as rough guidelines and are not necessarily indicative of the users actual performance.

ECS-sync can be scaled up or down by changing the number of sync servers. More sync servers allows for more jobs to be done simultaneously.

### General Performance ecs-sync 3.0 OVA With Validation

File size	No. Sync Servers	Jobs per Sync Server	No. of Access Nodes	Thread Count	24-hour Throughput/Server	Total/Day
1 - 3MB	2	1	2	30	7.6TB	15.2 TB
30KB	4	1	2	30	2.3M clip	9.2M Objects

### Real World Examples

Data Application	Avg. File Size	No. Sync Servers	No. of Access Nodes	24-hour Throughput/Server	Total/Day
PACS	1-3 MB	4	2	8TB	48TB
Enterprise Vault	300KB	8	2	1.5M Clips	12M Objects



---

### Installing the Service

---

The ecs-sync distribution comes with scripts to aid in configuring a Linux VM to run ecs-sync (and optionally the UI) as a service. In 3.0, this will be the standard configuration for all methods of running the tool (CLI, XML or Web UI). In 2.1, it is not required, but it is how [the OVA](#) is configured by default. Here is a rough diagram of the service architecture.

Fig. 2.1: ecs-sync 2.1 UI architecture

As you can see, the 2.1 UI was designed to support a single use-case of syncing Filesystem directories to ECS buckets. In 3.0, just about all storage plugins and filters will be supported as well. The goal for 3.0 is to allow any type of configuration to be submitted via CLI, XML or Web UI using essentially the same structure.

The installation scripts will install a number of dependencies, including Sqlite, MariaDB (OSS mySQL), Java, some standard analysis tools, etc. The procedure for preparing a VM and running these scripts is outlined below.

1. Create a 64-bit Linux VM (CentOS or RHEL 6.4+ is supported).
  - Ideally, the VM should have at least 8 vCores and 16GB memory, but the biggest requirement is to maximize the network pipe to both the source and the target.
  - Our internal OVA uses CentOS 7 minimal.
2. Upload the ecs-sync distribution zip and UI jar file to the VM.
  - Get these files from the latest release page [here](#).
  - Unzip the distribution zip (after which you can delete the zip file), then move the UI jar into the folder that was created (next to the ecs-sync jar).
3. Update installed packages to latest version: `sudo yum update`
4. Run the OS configuration script.
  - As root, change CWD to the distribution folder and run `ova/configure-centos.sh`.
  - At some point, you will be guided through the `mysql_secure_installation` script. There is no root password by default; you should set one.

- Remember your mySQL root password so that you can enter it when prompted later.
5. Run the ecs-sync install script.
    - This will install the ecs-sync and ecs-sync-ui services. If you are running 2.1 in CLI-only, you can skip this step.
    - As root, from within the distribution folder, run `ova/install.sh`.
  6. If you are performing any CAS migrations, install the [CAS SDK](#).
  7. If you are writing to a filesystem (NFS, SMB, etc.), install [iozone](#) to test write performance.
    - You should build an RPM from the SRPM, since there is no published package available for RedHat.
  8. If you are writing to ECS via S3, install [bucket-perf](#) to test S3 write performance.
    - This is just a self-executing jar file, so place it in an accessible location on the VM.

## Updates and vulnerabilities

Note that our pre-built OVA releases are based on CentOS minimal and, prior to release, are updated with the latest OS patches. However, there will always be some updates made available between release and deployment time. With this in mind, you are encouraged to always `sudo yum update` after deployment to avoid any vulnerability exposure in a production environment.



---

### Creating Config File

---

We recommend running `ecs-sync` by submitting an XML configuration file as a job. To this end we've built and included a new XML generator tool. This tool will generate a template configuration file to include the necessary plugins for your migration. This is done according to the options passed to it. Running the generator constructs an XML file that has all of the available options set to their defaults (if available).

#### Please note:

- This tool does not create a complete configuration, only a template that must be expanded and/or modified before it is ready to be run.
- The XML generator tool does not currently include error checking. This means that incorrect values, if passed into the generator, will also be passed into the newly created XML configuration file. **Please review the generated XML file before submitting it as a job to `ecs-sync` in order to ensure a successful migration.**
- Several sample configuration files are included in the repo under `ecs-sync/sample`. Please refer to these samples when building your configuration file.
- Some options will appear in the template with a value, but that value may not be valid. When generating, you can also optionally include comments describing each option and its possible values and default value.

#### The new generator is part of the `ecs-sync-ctl` tool and invoked like so:

```
usage: java -jar ecs-sync-ctl-{version}.jar --xml-gen <output-file>
       [--xml-comments] [--xml-source <source-prefix>] [--xml-filters
       <filter-list>] [--xml-target <target-prefix>]
--xml-comments           Adds descriptive comments to the
                           generated config file
--xml-filters <filter-list> A comma-delimited list of names of
                           filters to use as the source in the
                           generated config file (optional)
--xml-gen <output-file>  Generates a verbose XML config file
```

<code>--xml-source &lt;source-prefix&gt;</code>	<b>for</b> the specified plugins The prefix <b>for</b> the storage plugin to use <b>as</b> the source <b>in</b> the generated config file
<code>--xml-target &lt;target-prefix&gt;</code>	The prefix <b>for</b> the storage plugin to use <b>as</b> the target <b>in</b> the generated config file

Notice that XML Generator requires three arguments to run successfully. They are:

- Desired name of the xml file being created
- Necessary storage plugin for source
- Necessary storage plugin for target

Available storage plugins and their appropriate uses can be found [here](#).

For example `ecs-sync-ctl --xml-gen example.xml --xml-source s3 --xml-target ecs-s3` outputs the file `example.xml` for a sync **coming from** s3 type storage and **going to** ecs s3 type storage.

Example.xml sets the following options for the transfer:

```
<syncConfig xmlns="http://www.emc.com/ecs/sync/model">
  <options>
    <bufferSize>524288</bufferSize>
    <dbConnectionString>dbConnectionString</dbConnectionString>
    <dbFile>dbFile</dbFile>
    <dbTable>dbTable</dbTable>
    <deleteSource>false</deleteSource>
    <forceSync>false</forceSync>
    <ignoreInvalidAcls>false</ignoreInvalidAcls>
    <logLevel>quiet</logLevel>
    <monitorPerformance>true</monitorPerformance>
    <recursive>true</recursive>
    <rememberFailed>false</rememberFailed>
    <retryAttempts>2</retryAttempts>
    <sourceListFile>sourceListFile</sourceListFile>
    <syncAcl>false</syncAcl>
    <syncData>true</syncData>
    <syncMetadata>true</syncMetadata>
    <syncRetentionExpiration>false</syncRetentionExpiration>
    <threadCount>16</threadCount>
    <timingWindow>1000</timingWindow>
    <timingsEnabled>false</timingsEnabled>
    <verify>false</verify>
    <verifyOnly>false</verifyOnly>
  </options>
```

for the source:

```
<awsS3Config>
  <accessKey>accessKey</accessKey>
  <bucketName>bucketName</bucketName>
  <createBucket>false</createBucket>
  <decodeKeys>false</decodeKeys>
  <disableVHosts>false</disableVHosts>
  <host>host</host>
  <includeVersions>false</includeVersions>
```

```

    <keyPrefix>keyPrefix</keyPrefix>
    <legacySignatures>>false</legacySignatures>
    <mpuPartSizeMb>128</mpuPartSizeMb>
    <mpuThreadCount>4</mpuThreadCount>
    <mpuThresholdMb>512</mpuThresholdMb>
    <port>-1</port>
    <preserveDirectories>>false</preserveDirectories>
    <protocol>protocol</protocol>
    <secretKey>secretKey</secretKey>
    <socketTimeoutMs>50000</socketTimeoutMs>
  </awsS3Config>
</source>

```

and for the target:

```

<target>
  <ecsS3Config>
    <accessKey>accessKey</accessKey>
    <apacheClientEnabled>>false</apacheClientEnabled>
    <bucketName>bucketName</bucketName>
    <createBucket>>false</createBucket>
    <decodeKeys>>false</decodeKeys>
    <enableVHosts>>false</enableVHosts>
    <geoPinningEnabled>>false</geoPinningEnabled>
    <host>host</host>
    <includeVersions>>false</includeVersions>
    <keyPrefix>keyPrefix</keyPrefix>
    <mpuDisabled>>false</mpuDisabled>
    <mpuPartSizeMb>128</mpuPartSizeMb>
    <mpuThreadCount>4</mpuThreadCount>
    <mpuThresholdMb>512</mpuThresholdMb>
    <port>0</port>
    <preserveDirectories>>false</preserveDirectories>
    <protocol>protocol</protocol>
    <secretKey>secretKey</secretKey>
    <smartClientEnabled>true</smartClientEnabled>
    <socketConnectTimeoutMs>15000</socketConnectTimeoutMs>
    <socketReadTimeoutMs>60000</socketReadTimeoutMs>
    <vdcs>vdcs</vdcs>
  </ecsS3Config>
</target>

```

As noted previously, many fields, such as `accessKey`, `bucketName`, `protocol`, `port`, `secretKey`, etc. are set to placeholder values and must be changed accordingly depending on each specific case. **\*Without changing these placeholder values the configuration file cannot be run successfully\***. All values not filled with a placeholder are set to default values that may or may not apply to any particular case. **\*Be sure to review these values before submitting as a job\*** as they may need to be changed in order to fit your situation.



## CHAPTER 4

---

### Plugin Support

---

ecs-sync supports migrating to and from several different source and target file types. In order to achieve this the corresponding plugin is used. The appropriate plugin must be specified in the XML configuration file.

### Available plugins are listed below along with any custom options they may have

```
Archive File (archive:)
    The archive plugin reads/writes data from/to an archive file (tar, zip, etc.) It
    ↳ is
        triggered by an archive URL:
archive:[<scheme>://]<path>, e.g. archive:file:///home/user/myfiles.tar
or archive:http://company.com/bundles/project.tar.gz or archive:cwd_file.zip
The contents of the archive are the objects. To preserve object metadata on the target
    filesystem, or to read back preserved metadata, use --store-metadata.
    NOTE: Storage options must be prefixed by source- or target-, depending on which
    ↳ role
        they assume
        --delete-check-script <delete-check-script> when --delete-source is used, add
    ↳ this
                                                    option to execute an external
    ↳ script to
                                                    check whether a file should be
    ↳ deleted.
                                                    If the process exits with return
    ↳ code
                                                    zero, the file is safe to delete.
        --delete-older-than <delete-age> when --delete-source is used, add
    ↳ this
                                                    option to only delete files that
    ↳ have
                                                    been modified more than <delete-age>
                                                    milliseconds ago
```

<code>--excluded-paths &lt;pattern,pattern,...&gt;</code>	A list of regular expressions to
<code>↪search</code>	against the full file path. If the
<code>↪path</code>	matches, the file will be skipped.
<code>↪take</code>	Since this is a regular expression,
<code>↪For</code>	care to escape special characters.
<code>↪period,</code>	example, to exclude all files and
<code>    --follow-links</code>	directories that begin with a
<code>↪links,</code>	the pattern would be <code>./\..*</code>
<code>↪files</code>	instead of preserving symbolic
<code>    --modified-since &lt;yyyy-MM-ddThh:mm:ssZ&gt;</code>	follow them and sync the actual
<code>↪time.</code>	only look at files that have been
<code>↪8601</code>	modified since the specific date/
<code>↪01T04:30:00Z)</code>	Date/time should be provided in ISO-
<code>    --store-metadata</code>	UTC format (i.e. 2015-01-
<code>    --use-absolute-path</code>	when used as a target, stores source
<code>↪when</code>	metadata in a json file, since
<code>↪path</code>	filesystems have no concept of user
	metadata
	Uses the absolute path to the file
	storing it instead of the relative
	from the source dir
<b>Atmos (atmos:)</b>	
The Atmos plugin is triggered by the URI pattern:	
<code>atmos:http[s]://uid:secret@host[,host..][:port][/namespace-path]</code>	
Note that the uid should be the 'full token ID' including the subtenant ID and the uid concatenated by a slash	
If you want to software load balance across multiple hosts, you can provide a comma-delimited list of hostnames or IPs in the host part of the URI.	
NOTE: Storage options must be prefixed by source- or target-, depending on which	
<code>↪role</code>	
<code>    they assume</code>	
<code>    --access-type &lt;access-type&gt;</code>	The access method to locate objects
<code>    --preserve-object-id</code>	(objectspace or namespace)
<code>↪target</code>	Supported in ECS 3.0+ when used as a
<code>↪(both</code>	where another AtmosStorage is the source
<code>↪new ECS</code>	must use objectspace). When enabled, a
<code>↪legacy</code>	feature will be used to preserve the
<code>    --remove-tags-on-delete</code>	object ID, keeping all object IDs the same
	between the source and target
	When deleting from a source subtenant,
	specifies whether to delete listable-tags

↪done to	prior to deleting the object. This is
↪write	reduce the tag index size and improve
--replace-metadata	performance under the same tags Atmos does not have a call to replace metadata; only to set or remove it. By default, set is used, which means removed metadata will not be reflected when
↪updating	objects. Use this flag if your sync
↪operation	might remove metadata from an existing
↪object	
--ws-checksum-type <ws-checksum-type>	If specified, the atmos wschecksum feature will be applied to writes. Valid
↪algorithms	are sha1, or md5. Disabled by default
<b>S3 (s3:)</b>	
Represents storage in an Amazon S3 bucket. This plugin is triggered by the	
↪pattern:	
s3:[http[s]://]access_key:secret_key@[host[:port]]/bucket[/root-prefix]	
Scheme, host and port are all optional. If omitted, https://s3.amazonaws.com:443 is assumed. keyPrefix (optional) is the prefix under which to start enumerating or writing keys within the bucket, e.g. dir1/. If omitted, the root of the bucket is assumed.	
NOTE: Storage options must be prefixed by source- or target-, depending on which	
↪role	
they assume	
--create-bucket	By default, the target bucket must exist.
↪This	
--decode-keys	option will create it if it does not Specifies if keys will be URL-decoded
↪after	
↪you see	listing them. This can fix problems if
↪like	file or directory names with characters
--disable-v-hosts	%2f in them Specifies whether virtual hosted buckets
↪will	
↪be	be disabled (and path-style buckets will
--include-versions	used) Transfer all versions of every object.
↪NOTE:	
↪exist!	this will overwrite all versions of each source key in the target system if any
--legacy-signatures	Specifies whether the client will use v2
↪auth.	
--mpu-part-size-mb <size-in-MB>	Necessary for ECS < 3.0 Sets the part size to use when multipart upload is required (objects over 5GB).
↪Default	
--mpu-thread-count <mpu-thread-count>	is 128MB, minimum is 5MB The number of threads to use for multipart

```

--mpu-threshold-mb <size-in-MB>          upload (only applicable for file sources)
                                           Sets the size threshold (in MB) when an
↪upload
                                           shall become a multipart upload

--preserve-directories                    If enabled, directories are stored in S3
↪as
                                           empty objects to preserve empty dirs and
                                           metadata from the source

--socket-timeout-ms <timeout-ms>         Sets the socket timeout in milliseconds
                                           (default is 50000ms)

```

CAS (cas:)

The CAS plugin is triggered by the URI pattern:  
cas:[hpp:]/host[:port][,host[:port]...]?name=<name>,secret=<secret>  
or cas:[hpp:]/host[:port][,host[:port]...]?<pea\_file>  
Note that <name> should be of the format <subtenant\_id>:<uid> when connecting to an  
↪Atmos  
system. This is passed to the CAS SDK as the connection string (you can use  
↪primary=,  
secondary=, etc. in the server hints). To facilitate CAS migrations, sync from a  
CasStorage source to a CasStorage target. Note that by default, verification of a  
CasStorage object will also verify all blobs.  
NOTE: Storage options must be prefixed by source- or target-, depending on which  
↪role  
they assume  
--application-name <application-name> This is the application name given  
↪to  
the pool during initial connection.  
--application-version <application-version> This is the application version  
↪given to  
the pool during initial connection.  
--delete-reason <audit-string> When deleting source clips, this is  
↪the  
audit string.

ECS S3 (ecs-s3:)

Reads and writes content from/to an ECS S3 bucket. This plugin is triggered by the  
pattern:  
ecs-s3:http[s]://access\_key:secret\_key@hosts/bucket[/key-prefix] where hosts =  
host[,host][,..] or vdc-name(host,..)[,vdc-name(host,..)][,..] or load-  
↪balancer[:port]  
Scheme, host and port are all required. key-prefix (optional) is the prefix under  
↪which to  
start enumerating or writing within the bucket, e.g. dir1/. If omitted the root  
↪of the  
bucket will be enumerated or written to.  
NOTE: Storage options must be prefixed by source- or target-, depending on which  
↪role  
they assume  
--apache-client-enabled Enable this if you have disabled MPU  
↪and  
have objects larger than 2GB (the  
↪limit for  
the native Java HTTP client)  
--create-bucket By default, the target bucket must  
↪exist.  
This option will create it if it does  
↪not



<code>--decode-keys</code>	Specifies if keys will be URL-decoded
<code>↪after</code>	listing them. This can fix problems if
<code>↪you</code>	see file or directory names with
<code>↪characters</code>	like %2f in them
<code>--enable-v-hosts</code>	Specifies whether virtual hosted
<code>↪buckets</code>	will be used (default is path-style buckets)
<code>--geo-pinning-enabled</code>	Enables geo-pinning. This will use a standard algorithm to select a
<code>↪consistent</code>	VDC for each object key or bucket name
<code>--include-versions</code>	Enable to transfer all versions of
<code>↪every</code>	object. NOTE: this will overwrite all versions of each source key in the
<code>↪target</code>	system if any exist!
<code>--mpu-disabled</code>	Disables multi-part upload (MPU). Large files will be sent in a single stream
<code>--mpu-part-size-mb &lt;size-in-MB&gt;</code>	Sets the part size to use when
<code>↪multipart</code>	upload is required (objects over 5GB). Default is 128MB, minimum is 4MB
<code>--mpu-thread-count &lt;mpu-thread-count&gt;</code>	The number of threads to use for
<code>↪multipart</code>	upload (only applicable for file
<code>↪sources)</code>	Sets the size threshold (in MB) when an upload shall become a multipart upload
<code>--mpu-threshold-mb &lt;size-in-MB&gt;</code>	The smart-client is enabled by default.
<code>--no-smart-client</code>	this option to turn it off when using a load balancer or fixed set of nodes
<code>↪ Use</code>	If enabled, directories are stored in
<code>--preserve-directories</code>	empty objects to preserve empty dirs
<code>↪S3 as</code>	metadata from the source
<code>↪and</code>	Sets the connection timeout in
<code>--socket-connect-timeout-ms &lt;timeout-ms&gt;</code>	(default is 15000ms)
<code>↪milliseconds</code>	Sets the read timeout in milliseconds (default is 60000ms)
<code>--socket-read-timeout-ms &lt;timeout-ms&gt;</code>	
Filesystem (file:)	
The filesystem plugin reads/writes data from/to a file or directory. It is	
<code>↪triggered</code>	by the URI:
file://<path>, e.g. file:///home/user/myfiles	
If the URL refers to a file, only that file will be synced. If a directory is	
<code>↪specified,</code>	the contents of the directory will be synced. Unless the <code>--non-recursive</code> flag is
<code>↪set,</code>	the subdirectories will also be recursively synced. To preserve object metadata
<code>↪on the</code>	

```

target filesystem, or to read back preserved metadata, use --store-metadata.
NOTE: Storage options must be prefixed by source- or target-, depending on which
role
they assume
--delete-check-script <delete-check-script> when --delete-source is used, add
this
option to execute an external
script to
check whether a file should be
deleted.
If the process exits with return
code
zero, the file is safe to delete.
--delete-older-than <delete-age> when --delete-source is used, add
this
option to only delete files that
have
been modified more than <delete-age>
milliseconds ago
--excluded-paths <pattern,pattern,...> A list of regular expressions to
search
against the full file path. If the
path
matches, the file will be skipped.
Since this is a regular expression,
take
care to escape special characters.
For
example, to exclude all files and
directories that begin with a
period,
the pattern would be .*/\.*
instead of preserving symbolic
links,
follow them and sync the actual
files
only look at files that have been
modified since the specifiec date/
time.
Date/time should be provided in ISO-
8601
UTC format (i.e. 2015-01-
01T04:30:00Z)
--store-metadata
when used as a target, stores source
metadata in a json file, since
filesystems have no concept of user
metadata
--use-absolute-path
Uses the absolute path to the file
when
storing it instead of the relative
path
from the source dir

Simulated Storage for Testing (test:)
This plugin will generate random data when used as a source, or act as /dev/null
when
used as a target
NOTE: Storage options must be prefixed by source- or target-, depending on which
role

```

they assume	
--chance-of-children <chance-of-children>	When used as a source, the percent
↳chance	
↳data	that an object is a directory vs a
--max-child-count <max-child-count>	object. Default is 30
↳child	When used as a source, the maximum
↳count	count for a directory (actual child
--max-depth <max-depth>	is random). Default is 8
↳is 5	When used as a source, the maximum
--max-metadata <max-metadata>	of metadata tags to generate (actual
↳number	number is random). Default is 5
--max-size <max-size>	When used as a source, the maximum
↳size of	objects (actual size is random).
↳Default	is 1048576
--no-discard-data	By default, all data generated or read
↳store	will be discarded. Turn this off to
--object-count <object-count>	the object data and index in memory
↳number of	When used as a source, the exact
↳100	root objects to generate. Default is
--object-owner <object-owner>	When used as a source, specifies the
↳owner	of every object (in the ACL)
--read-data	When used as a target, actually read
↳the	data from the source (data is not
↳read by	default)
--valid-groups <valid-groups>	When used as a source, specifies valid
↳grants	groups for which to generate random
--valid-permissions <valid-permissions>	in the ACL
↳random	When used as a source, specifies valid
--valid-users <valid-users>	permissions to use when generating
↳grants	grants
	When used as a source, specifies valid
	users for which to generate random
	in the ACL

#### ACL Mapper (acl-mapping)

The ACL Mapper will map ACLs from the source system to the target using a provided mapping file. The mapping file should be ordered by priority and will short-

↳circuit

(the first mapping found for the source key will be chosen for the target). Note

↳that

if a mapping is not specified for a user/group/permission, that value will remain unchanged in the ACL of the object. You can optionally remove grants by leaving

the target value empty and you can add grants to all objects using the `--acl-add-grants` option.

If you wish to migrate ACLs with your data, you will always need this plugin unless the users, groups and permissions in both systems match exactly. Note: If you simply want to take the default ACL of the target system, there is no need for this filter; just don't sync ACLs (this is the default behavior)

`--acl-add-grants <acl-add-grants>` Adds a comma-separated list of grants to all objects synced to the target system.

Syntax is like so (repeats are allowed):

`<target_group>=<target_perm>,user.get_user=<target_perm>`

`--acl-append-domain <acl-append-domain>` Appends a directory realm/domain to each user that is mapped. Useful when mapping POSIX users to LDAP identities

`--acl-map-file <acl-map-file>` Path to a file that contains the identities and permissions from source target. Each entry is on a separate line and specifies a group/user/permission and target name[s] like so:

`group.<source_group>=<target_group>`  
`user.<source_user>=<target_user>`  
`permission.<source_perm>=<target_perm>[,get_perm>..]`

You can also pare down permissions that redundant in the target system by using permission groups. I.e.:

`permission1.WRITE=READ_WRITE`  
`permission1.READ=READ`  
 will pare down separate READ and WRITE permissions into one `READ_WRITE/READ` the ordering by priority). Groups are processed before straight mappings.

the target value blank to flag an identity/permission that should be removed (perhaps it does not exist in the target system)

`--acl-strip-domain` Strips the directory realm/domain from each user that is mapped. Useful when mapping

```

--acl-strip-groups          LDAP identities to POSIX users
↪ Use                      Drops all groups from each object's ACL.
                             with --acl-add-grants to add specific_
↪ group                    grants instead
                             Drops all users from each object's ACL._
--acl-strip-users          ↪ Use
                             with --acl-add-grants to add specific_
↪ user                    grants instead

Decryption Filter (decrypt)
  Decrypts object data using the Atmos Java SDK encryption standard
  (https://community.emc.com/docs/DOC-34465). This method uses envelope encryption_
↪ where
  each object has its own symmetric key that is itself encrypted using the master
  asymmetric key. As such, there are additional metadata fields added to the object_
↪ that
  are required for decrypting
  --decrypt-keystore <keystore-file>          required. the .jks keystore file_
↪ that                                       holds the decryption keys. which_
↪ key to                                   use is actually stored in the object
                                           metadata
                                           the keystore password
                                           by default, the modification time
                                           (mtime) of an object does not change
                                           when decrypted. set this flag to_
--decrypt-keystore-pass <keystore-password>
--decrypt-update-mtime
↪ update
                                           the mtime. useful for in-place
                                           decryption when objects would not
                                           otherwise be overwritten due to_
↪ matching
                                           timestamps
                                           by default, if an object is not
                                           encrypted, it will be passed_
--fail-if-not-encrypted
↪ through the
                                           filter chain untouched. set this_
↪ flag to
                                           fail the object if it is not_
↪ encrypted

Encryption Filter (encrypt)
  Encrypts object data using the Atmos Java SDK encryption standard
  (https://community.emc.com/docs/DOC-34465). This method uses envelope encryption_
↪ where
  each object has its own symmetric key that is itself encrypted using the master
  asymmetric key. As such, there are additional metadata fields added to the object_
↪ that
  are required for decrypting. Note that currently, metadata is not encrypted
  --encrypt-force-strong          256-bit cipher strength is always_
↪ used
                                           if available. this option will stop
                                           operations if strong ciphers are not
                                           available
                                           the alias of the master encryption_
--encrypt-key-alias <encrypt-key-alias>
↪ key

```

<code>--encrypt-keystore &lt;keystore-file&gt;</code>	within the keystore
↳the	the .jks keystore file that holds
<code>--encrypt-keystore-pass &lt;keystore-password&gt;</code>	master encryption key
<code>--encrypt-update-mtime</code>	the keystore password
↳update	by default, the modification time (mtime) of an object does not change when encrypted. set this flag to
↳matching	the mtime. useful for in-place encryption when objects would not otherwise be overwritten due to
<code>--fail-if-encrypted</code>	timestamps
↳will be	by default, if an object is already encrypted using this method, it
	passed through the filter chain untouched. set this flag to fail the object if it is already encrypted
<b>Gladinet Mapper (gladinet-mapping)</b>	
This plugin creates the appropriate metadata in Atmos to upload data in a fashion compatible with Gladinet's Cloud Desktop software when it's hosted by EMC Atmos	
<code>--gladinet-dir &lt;base-directory&gt;</code>	Sets the base directory in Gladinet to load
↳content	into. This directory must already exist
<b>ID Logging Filter (id-logging)</b>	
Logs the input and output Object IDs to a file. These IDs are specific to the	
↳source	and target plugins
<code>--id-log-file &lt;path-to-file&gt;</code>	The path to the file to log IDs to
<b>Local Cache (local-cache)</b>	
Writes each object to a local cache directory before writing to the target.	
↳Useful for	applying external transformations or for transforming objects in-place (source/
↳target	are the same)
<b>NOTE:</b> this filter will remove any extended properties from storage plugins (i.e.	
↳versions,	CAS tags, etc.) Do not use this plugin if you are using those features
<code>--local-cache-root &lt;cache-directory&gt;</code>	specifies the root directory in which to
↳cache	files
<b>Metadata Filter (metadata)</b>	
Allows adding regular and listable (Atmos only) metadata to each object	
<code>--add-listable-metadata &lt;name=value,name=value,...&gt;</code>	Adds listable metadata to
↳every	object
<code>--add-metadata &lt;name=value,name=value,...&gt;</code>	Adds regular metadata to
↳every	object
<b>Override Mimetype (override-mimetype)</b>	
This plugin allows you to override the default mimetype of objects getting	

transferred. It is useful for instances where the mimetype of an object cannot be inferred from its extension or is nonstandard (not in Java's mime.types file).

→ You can also use the force option to override the mimetype of all objects

<code>--force-mimetype</code>	If specified, the mimetype will be overwritten regardless of its prior value
<code>--override-mimetype &lt;mimetype&gt;</code>	Specifies the mimetype to use when an object has

→ no default mimetype

**Preserve ACLs (preserve-acl)**  
This plugin will preserve source ACL information as user metadata on each object

**Preserve File Attributes (preserve-file-attributes)**  
This plugin will read and preserve POSIX file attributes as metadata on the object

**Restore Preserved ACLs (restore-acl)**  
This plugin will read preserved ACLs from user metadata and restore them to each object

**Restore File Attributes (restore-file-attributes)**  
This plugin will restore POSIX file attributes that were previously preserved in metadata on the object

**Shell Command Filter (shell-command)**  
Executes a shell command after each successful transfer. The command will be

→ given two arguments: the source identifier and the target identifier

<code>--shell-command &lt;path-to-command&gt;</code>	The shell command to execute
--	------------------------------





## CHAPTER 5

---

### Initializing the UI

---

(this page applies to ecs-sync 3.1+)

#### Quick Start

To quickly start using the ecs-sync UI, load the home page in a browser (`https://{vm_ip}`). The default login is `admin/ecs-sync`. On first login, type in an alert email address (if you won't use scheduling or alerts, this doesn't have to be real). Then click **Save & Write Configuration to Storage**. That's it! The UI is now ready to use.



# CHAPTER 6

---

## Getting Started

---

Ecs-sync requires an XML configuration file in order to run a sync. Previously these had to be written by hand, or by using the XML generator. While these are still legitimate options for running ecs-sync, there is now a simpler, faster, and easier way.

The new ecs-sync UI has been released, making running migrations simpler than ever before. The following guide will lay out instructions for its use. With the addition of the new web UI it is no longer necessary to run migrations through the command line or manually create, or edit, the required XML config files

## Installing the UI

The new UI is installed easily and instructions can be found [here](#) with the general ecs-sync instructions.

## Logging In

The default credentials for the sync UI are admin/ecs-sync. It is, of course, recommended that this password be changed immediately. The default password is changed by running `sudo htpasswd /etc/httpd/.htpasswd admin` on the ecs-sync VM.

After a fresh installation the UI must be initialized before it can be used. Upon first login the user will see:

This means that the UI must be initialized before being used. As noted in the troubleshooting page, the grails error can be ignored and will resolve once a user email is successfully submitted. Instructions can be found [here](#).

Note the option to store configuration files on a remote ECS server. This is a helpful way of preserving configuration files independently of ecs-sync servers, in the case of teardown, rebuild, etc.

## Starting a sync

After the UI is initialized it is ready to be used. A single, one-time sync can be run by going to the ‘Status’ tab on the top left. This will show the user any currently active jobs, a “New Sync” button, and basic user statistics. Clicking the “New Sync” shows source, target, and sync options fields. Source and target are used to select the appropriate plugins for the sync. Selecting the desired plugins yields: As you can see the UI prompts the user for the information required for each plugin to function. It may be necessary to change default settings such as port number, VDCs, etc. This is done by clicking on “show advanced options.” Be sure to take a look at these options before starting your sync, as some may be necessary.

## Filters

Note that filters will be applied in the order specified, so make sure the order is appropriate. For example, a source-extraction filter should come before a target-ingest filter.

## Sync Options

While these fields are optional, they can prove to be very important. Object list, Verify, and Thread Count, are all important options that should be considered before running your sync

## Database Table

Particular attention must be paid to the field “Db Table.” Ecs-sync records every sync in a database table that is available for later review. However, if this field is left blank, that is the database remains unnamed, ecs-sync will consider the table **temporary** and **\*the table will be wiped on completion\***. If the table is named before the sync, it will be retained until manually wiped. This is important to note as the table may be necessary to the user at a later time. Please keep this in mind for every sync.

## CHAPTER 7

### CLI Syntax

The following is the complete syntax of the CLI arguments for 3.0. Note that you can also generate this text simply by running: `java -jar ecs-sync-3.0.jar --help`

Full 3.0 CLI syntax:

```
EcsSync v3.0
usage: java -jar ecs-sync.jar -source <source-uri> [-filters <filter1>[,<filter2>,...
↪]]
        -target <target-uri> [options]
Common options:
    --buffer-size <buffer-size>          Sets the buffer size (in bytes) to use
↪when
                                           streaming data from the source to the
↪target
                                           (supported plugins only). Defaults to
↪512K
    --db-connect-string <db-connect-string> Enables the MySQL database engine and
↪connect
                                           specified the JDBC connect string to
                                           to the database (i.e.
↪user=f
                                           "jdbc:mysql://localhost:3306/ecs_sync?
                                           oo&password=bar")
    --db-file <db-file>                  Enables the Sqlite database engine and
↪runs
                                           specifies the file to hold the status
                                           database. A database will make repeat
                                           and incrementals more efficient. You can
↪interrogate
                                           also use the sqlite3 client to
                                           the details of all objects in the sync
    --db-table <db-table>                Specifies the DB table name to use. Use
↪this
                                           with --db-connect-string to provide a
↪unique
```

<pre> ↳ previously     --delete-source ↳ each ↳ synced  ↳ (source     --filters &lt;filter-names&gt; ↳ apply ↳ the ↳ [returned ↳ Examples:      --force-sync ↳ regardless      --help     --ignore-invalid-acls ↳ e.  ↳ exist      --log-level &lt;log-level&gt;      --no-monitor-performance ↳ reads and ↳ This      --no-rest-server     --no-sync-data     --no-sync-metadata     --non-recursive ↳ recursively      --perf-report-seconds &lt;seconds&gt; ↳ seconds      --remember-failed      --rest-endpoint &lt;rest-endpoint&gt; ↳ the </pre>	<pre> table name or risk corrupting a used table. Default table is "objects" Supported source plugins will delete source object once it is successfully (does not include directories). Use this option with care! Be sure log levels are appropriate to capture transferred deleted) objects The comma-delimited list of filters to to objects as they are synced. Specify activation names of the filters from Filter.getActivationName()]. id-logging gladinet-mapping,strip-acls Each filter may have additional custom parameters you may specify separately Force the write of each object, of its state in the target storage Displays this help content If syncing ACL information when syncing objects, ignore any invalid entries (i. permissions or identities that don't in the target system) Sets the verbosity of logging (silent quiet verbose debug). Default is quiet Enables performance monitoring for writes on any plugin that supports it. information is available via the REST service during a sync Disables the REST server Object data is synced by default Metadata is synced by default Hierarchical storage will sync by default Report upload and download rates for the source and target plugins every &lt;x&gt; to INFO logging. Default is off (0) Tracks all failed objects and displays a summary of failures when finished Specified the host and port to use for REST endpoint. Optional; defaults to </pre>
---	--

```

--rest-only
↳start

--retry-attempts <retry-attempts>
↳should

--source <source-uri>
↳Examples:

↳be

--source-list-file <source-list-file>
↳be in
↳and the
↳line.
↳plugin

--sync-acl
↳objects

--sync-retention-expiration
↳when
↳The
↳replicate
↳Works

↳an
↳enable
↳this to

--target <target-uri>
↳Examples:

↳be

```

localhost:9200  
Enables REST-only control. This will

the REST server and remain alive until manually terminated. Excludes all other options except --rest-endpoint  
Specifies how many times each object

be retried after an error. Default is 2 retries (total of 3 attempts)  
The URI for the source storage.

atmos:http://uid:secret@host:port  
'- Uses Atmos as the source; could also

https.  
file:///tmp/atmos/  
'- Reads from a directory  
archive:///tmp/atmos/backup.tar.gz  
'- Reads from an archive file  
s3:http://key:secret@host:port  
'- Reads from an S3 bucket  
Other plugins may be available. See

documentation for URI formats  
Path to a file that supplies the list of source objects to sync. This file must

CSV format, with one object per line

identifier is the first value in each

This entire line is available to each

as a raw string  
Sync ACL information when syncing

(in supported plugins)  
Sync retention/expiration information

syncing objects (in supported plugins).

target plugin will \*attempt\* to

retention/expiration for each object.

only on plugins that support retention/expiration. If the target is

Atmos cloud, the target policy must

retention/expiration immediately for

work  
The URI for the target storage.

atmos:http://uid:secret@host:port  
'- Uses Atmos as the target; could also

```

https.
file:///tmp/atmos/
'- Writes to a directory
archive:///tmp/atmos/backup.tar.gz
'- Writes to an archive file
s3:http://key:secret@host:port
'- Writes to an S3 bucket
Other plugins may be available. See
↳their
--thread-count <thread-count>
--timing-window <timing-window>
↳Every
--timings-enabled
↳ins
--verify
--verify-only
↳object
↳does not
--verify-only
↳object
↳operations
--version
--xml-config <xml-config>
↳this
↳ignored.

```

documentation for URI formats  
Specifies the number of objects to sync simultaneously. Default is 16  
Sets the window for timing statistics.  
{timingWindow} objects that are synced, timing statistics are logged and reset. Default is 10,000 objects  
Enables operation timings on all plugins that support it  
After a successful object transfer, the object will be read back from the target system and its MD5 checksum will be compared with that of the source object during transfer). This only compares data (metadata is not compared) and include directories  
Similar to --verify except that the transfer is skipped and only reads are performed (no data is written)  
Displays package version  
Specifies an XML configuration file. In mode, the XML file contains all of the configuration for the sync job. In this mode, most other CLI arguments are

Available plugins are listed below along with any custom options they may have

#### Archive File (archive:)

The archive plugin reads/writes data from/to an archive file (tar, zip, etc.) It is triggered by an archive URL:  
archive:[<scheme>://]<path>, e.g. archive:file:///home/user/myfiles.tar  
or archive:http://company.com/bundles/project.tar.gz or archive:cwd\_file.zip  
The contents of the archive are the objects. To preserve object metadata on the target filesystem, or to read back preserved metadata, use --store-metadata.  
NOTE: Storage options must be prefixed by source- or target-, depending on which role they assume  
--delete-check-script <delete-check-script> when --delete-source is used, add this



```

↪script to
↪deleted.
↪code
    --delete-older-than <delete-age>
↪this
↪have
    --excluded-paths <pattern,pattern,...>
↪search
↪path
↪take
↪For
↪period,
    --follow-links
↪links,
↪files
    --modified-since <yyyy-MM-ddThh:mm:ssZ>
↪time.
↪8601
↪01T04:30:00Z)
    --store-metadata
↪
    --use-absolute-path
↪when
↪path

```

option to execute an external

check whether a file should be

If the process exits with return

zero, the file is safe to delete.  
when `--delete-source` is used, add

option to only delete files that

been modified more than `<delete-age>`  
milliseconds ago  
A list of regular expressions to

against the full file path. If the

matches, the file will be skipped.  
Since this is a regular expression,

care to escape special characters.

example, to exclude all files and  
directories that begin with a

the pattern would be `.*\/\..*`  
instead of preserving symbolic

follow them and sync the actual

only look at files that have been  
modified since the specific date/

Date/time should be provided in ISO-  
UTC format (i.e. 2015-01-

when used as a target, stores source  
metadata in a json file, since  
filesystems have no concept of user  
metadata  
Uses the absolute path to the file

storing it instead of the relative

from the source dir

Atmos (atmos:)

The Atmos plugin is triggered by the URI pattern:

```
atmos:http[s]://uid:secret@host[,host..][:port][/namespace-path]
```

Note that the uid should be the 'full token ID' including the subtenant ID and the uid concatenated by a slash

If you want to software load balance across multiple hosts, you can provide a comma-delimited list of hostnames or IPs in the host part of the URI.

NOTE: Storage options must be prefixed by `source-` or `target-`, depending on which

```

↪role
    they assume
    --access-type <access-type>

```

The access method to locate objects

<code>--preserve-object-id</code>	(objectspace or namespace)
<code>↪target</code>	Supported in ECS 3.0+ when used as a
<code>↪(both</code>	where another AtmosStorage is the source
<code>↪new ECS</code>	must use objectspace). When enabled, a
<code>↪legacy</code>	feature will be used to preserve the
<code>--remove-tags-on-delete</code>	object ID, keeping all object IDs the same between the source and target
<code>↪done to</code>	When deleting from a source subtenant, specifies whether to delete listable-tags prior to deleting the object. This is
<code>↪write</code>	reduce the tag index size and improve
<code>--replace-metadata</code>	performance under the same tags
<code>↪updating</code>	Atmos does not have a call to replace metadata; only to set or remove it. By default, set is used, which means removed metadata will not be reflected when
<code>↪operation</code>	objects. Use this flag if your sync
<code>↪object</code>	might remove metadata from an existing
<code>--ws-checksum-type &lt;ws-checksum-type&gt;</code>	If specified, the atmos wschecksum feature will be applied to writes. Valid
<code>↪algorithms</code>	are sha1, or md5. Disabled by default
<b>S3 (s3:)</b>	
Represents storage in an Amazon S3 bucket. This plugin is triggered by the	
<code>↪pattern:</code>	
<code>s3:[http[s]://]access_key:secret_key@[host[:port]]/bucket[/root-prefix]</code>	
Scheme, host and port are all optional. If omitted, <code>https://s3.amazonaws.com:443</code> is assumed. <code>keyPrefix</code> (optional) is the prefix under which to start enumerating or writing keys within the bucket, e.g. <code>dir1/</code> . If omitted, the root of the bucket is assumed.	
NOTE: Storage options must be prefixed by <code>source-</code> or <code>target-</code> , depending on which	
<code>↪role</code>	
they assume	
<code>--create-bucket</code>	By default, the target bucket must exist.
<code>↪This</code>	
<code>--decode-keys</code>	option will create it if it does not
<code>↪after</code>	Specifies if keys will be URL-decoded
<code>↪you see</code>	listing them. This can fix problems if
<code>↪like</code>	file or directory names with characters
<code>--disable-v-hosts</code>	<code>%2f</code> in them
<code>↪will</code>	Specifies whether virtual hosted buckets
<code>↪be</code>	be disabled (and path-style buckets will
	used)

```

--include-versions          Transfer all versions of every object.
↪NOTE:                      this will overwrite all versions of each
                              source key in the target system if any
↪exist!
  --legacy-signatures       Specifies whether the client will use v2
↪auth.                      Necessary for ECS < 3.0
                              Sets the part size to use when multipart
                              upload is required (objects over 5GB).
  --mpu-part-size-mb <size-in-MB>
↪Default                    is 128MB, minimum is 5MB
                              The number of threads to use for multipart
                              upload (only applicable for file sources)
  --mpu-thread-count <mpu-thread-count>
                              Sets the size threshold (in MB) when an
↪upload                     shall become a multipart upload
                              If enabled, directories are stored in S3
  --preserve-directories    empty objects to preserve empty dirs and
↪as                          metadata from the source
                              Sets the socket timeout in milliseconds
                              (default is 50000ms)
  --socket-timeout-ms <timeout-ms>

```

#### CAS (cas:)

The CAS plugin is triggered by the URI pattern:

```
cas:[hpp:]/host[:port][,host[:port]...]?name=<name>,secret=<secret>
```

```
or cas:[hpp:]/host[:port][,host[:port]...]?<pea_file>
```

Note that <name> should be of the format <subtenant\_id>:<uid> when connecting to an

```

↪Atmos
  system. This is passed to the CAS SDK as the connection string (you can use
↪primary=,
  secondary=, etc. in the server hints). To facilitate CAS migrations, sync from a
  CasStorage source to a CasStorage target. Note that by default, verification of a
  CasStorage object will also verify all blobs.
  NOTE: Storage options must be prefixed by source- or target-, depending on which
↪role
  they assume
  --application-name <application-name>      This is the application name given
↪to                                           the pool during initial connection.
  --application-version <application-version> This is the application version
↪given to                                   the pool during initial connection.
  --delete-reason <audit-string>             When deleting source clips, this is
↪the                                         audit string.

```

#### ECS S3 (ecs-s3:)

Reads and writes content from/to an ECS S3 bucket. This plugin is triggered by the pattern:

```
ecs-s3:http[s]://access_key:secret_key@hosts/bucket[/key-prefix] where hosts =
  host[,host][,...] or vdc-name(host,...)[,vdc-name(host,...)][,...] or load-
↪balancer[:port]
```

Scheme, host and port are all required. key-prefix (optional) is the prefix under

```

↪which to
  start enumerating or writing within the bucket, e.g. dir1/. If omitted the root
↪of the

```

```

bucket will be enumerated or written to.
NOTE: Storage options must be prefixed by source- or target-, depending on which
role
they assume
--apache-client-enabled          Enable this if you have disabled MPU
and
have objects larger than 2GB (the
limit for
the native Java HTTP client)
--create-bucket                  By default, the target bucket must
exist.
This option will create it if it does
not
--decode-keys                    Specifies if keys will be URL-decoded
after
listing them. This can fix problems if
you
see file or directory names with
characters
like %2f in them
--enable-v-hosts                Specifies whether virtual hosted
buckets
will be used (default is path-style
buckets)
--geo-pinning-enabled           Enables geo-pinning. This will use a
consistent
standard algorithm to select a
VDC for each object key or bucket name
--include-versions              Enable to transfer all versions of
every
object. NOTE: this will overwrite all
versions of each source key in the
target
system if any exist!
--mpu-disabled                  Disables multi-part upload (MPU). Large
files will be sent in a single stream
--mpu-part-size-mb <size-in-MB> Sets the part size to use when
multipart
upload is required (objects over 5GB).
Default is 128MB, minimum is 4MB
--mpu-thread-count <mpu-thread-count> The number of threads to use for
multipart
upload (only applicable for file
sources)
--mpu-threshold-mb <size-in-MB> Sets the size threshold (in MB) when an
upload shall become a multipart upload
--no-smart-client               The smart-client is enabled by default.
Use
this option to turn it off when using a
load balancer or fixed set of nodes
--preserve-directories          If enabled, directories are stored in
S3 as
empty objects to preserve empty dirs
and
metadata from the source
--socket-connect-timeout-ms <timeout-ms> Sets the connection timeout in
milliseconds
(default is 15000ms)

```

```

--socket-read-timeout-ms <timeout-ms>      Sets the read timeout in milliseconds
                                              (default is 60000ms)

Filesystem (file:)
  The filesystem plugin reads/writes data from/to a file or directory. It is
  triggered
  by the URI:
file://<path>, e.g. file:///home/user/myfiles
If the URL refers to a file, only that file will be synced. If a directory is
specified,
  the contents of the directory will be synced. Unless the --non-recursive flag is
set,
  the subdirectories will also be recursively synced. To preserve object metadata
on the
  target filesystem, or to read back preserved metadata, use --store-metadata.
NOTE: Storage options must be prefixed by source- or target-, depending on which
role
  they assume
  --delete-check-script <delete-check-script> when --delete-source is used, add
this
  option to execute an external
script to
  check whether a file should be
deleted.
  If the process exits with return
code
  zero, the file is safe to delete.
  when --delete-source is used, add
this
  option to only delete files that
have
  been modified more than <delete-age>
milliseconds ago
  A list of regular expressions to
search
  against the full file path. If the
path
  matches, the file will be skipped.
  Since this is a regular expression,
take
  care to escape special characters.
For
  example, to exclude all files and
directories that begin with a
period,
  the pattern would be .*/\.*
  instead of preserving symbolic
links,
  follow them and sync the actual
files
  only look at files that have been
  modified since the specified date/
time.
  Date/time should be provided in ISO-
8601
  UTC format (i.e. 2015-01-
01T04:30:00Z)
  --store-metadata when used as a target, stores source

```

<code>--use-absolute-path</code>	metadata in a json file, since filesystems have no concept of user metadata
<code>↪when</code>	Uses the absolute path to the file
<code>↪path</code>	storing it instead of the relative
	from the source dir
Simulated Storage for Testing (test:)	
This plugin will generate random data when used as a source, or act as /dev/null	
<code>↪when</code>	
used as a target	
NOTE: Storage options must be prefixed by source- or target-, depending on which	
<code>↪role</code>	
they assume	
<code>--chance-of-children &lt;chance-of-children&gt;</code>	When used as a source, the percent
<code>↪chance</code>	that an object is a directory vs a
<code>↪data</code>	object. Default is 30
<code>--max-child-count &lt;max-child-count&gt;</code>	When used as a source, the maximum
<code>↪child</code>	count for a directory (actual child
<code>↪count</code>	is random). Default is 8
<code>--max-depth &lt;max-depth&gt;</code>	When used as a source, the maximum
<code>↪is 5</code>	directory depth for children. Default
<code>--max-metadata &lt;max-metadata&gt;</code>	When used as a source, the maximum
<code>↪number</code>	of metadata tags to generate (actual
<code>--max-size &lt;max-size&gt;</code>	number is random). Default is 5
<code>↪size of</code>	When used as a source, the maximum
<code>↪Default</code>	objects (actual size is random).
<code>--no-discard-data</code>	is 1048576
<code>↪store</code>	By default, all data generated or read
<code>--object-count &lt;object-count&gt;</code>	will be discarded. Turn this off to
<code>↪number of</code>	the object data and index in memory
<code>↪100</code>	When used as a source, the exact
<code>--object-owner &lt;object-owner&gt;</code>	root objects to generate. Default is
<code>↪owner</code>	When used as a source, specifies the
<code>--read-data</code>	of every object (in the ACL)
<code>↪the</code>	When used as a target, actually read
<code>↪read by</code>	data from the source (data is not
<code>--valid-groups &lt;valid-groups&gt;</code>	default)
<code>↪grants</code>	When used as a source, specifies valid
	groups for which to generate random
	in the ACL

<code>--valid-permissions &lt;valid-permissions&gt;</code>	When used as a source, specifies valid permissions to use when generating
<code>↪random</code>	
<code>--valid-users &lt;valid-users&gt;</code>	grants
<code>↪grants</code>	When used as a source, specifies valid users for which to generate random
	in the ACL
<b>ACL Mapper (acl-mapping)</b>	
The ACL Mapper will map ACLs from the source system to the target using a provided mapping file. The mapping file should be ordered by priority and will short-	
<code>↪circuit</code>	(the first mapping found for the source key will be chosen for the target). Note
<code>↪that</code>	if a mapping is not specified for a user/group/permission, that value will remain unchanged in the ACL of the object. You can optionally remove grants by leaving
<code>↪the</code>	target value empty and you can add grants to all objects using the <code>--acl-add-</code>
<code>↪grants</code>	option.
If you wish to migrate ACLs with your data, you will always need this plugin unless	
<code>↪the</code>	users, groups and permissions in both systems match exactly. Note: If you simply
<code>↪want</code>	to take the default ACL of the target system, there is no need for this filter;
<code>↪just</code>	don't sync ACLs (this is the default behavior)
<code>--acl-add-grants &lt;acl-add-grants&gt;</code>	Adds a comma-separated list of grants
<code>↪to all</code>	objects synced to the target system.
<code>↪Syntax</code>	is like so (repeats are allowed):
<code>↪&lt;tar</code>	group.<target_group>=<target_perm>,user.
<code>--acl-append-domain &lt;acl-append-domain&gt;</code>	get_user>=<target_perm>
<code>--acl-map-file &lt;acl-map-file&gt;</code>	Appends a directory realm/domain to each user that is mapped. Useful when mapping
<code>↪mapping of</code>	POSIX users to LDAP identities
<code>↪to</code>	Path to a file that contains the
<code>↪line</code>	identities and permissions from source
<code>↪source</code>	target. Each entry is on a separate
<code>↪&lt;tar</code>	and specifies a group/user/permission
<code>↪are</code>	and target name[s] like so:
	group.<source_group>=<target_group>
	user.<source_user>=<target_user>
	permission.<source_perm>=<target_perm>[,
	get_perm>..]
	You can also pare down permissions that
	redundant in the target system by using
	permission groups. I.e.:
	permission1.WRITE=READ_WRITE

```

permission1.READ=READ
will pare down separate READ and WRITE
permissions into one READ_WRITE/READ_
↳ (note
↳ Leave
↳ removed
    --acl-strip-domain
↳ each
    --acl-strip-groups
↳ Use
↳ group
    --acl-strip-users
↳ Use
↳ user
    permission1.READ=READ
    will pare down separate READ and WRITE
    permissions into one READ_WRITE/READ_
    the ordering by priority). Groups are
    processed before straight mappings._
    the target value blank to flag an
    identity/permission that should be_
    (perhaps it does not exist in the target
    system)
    Strips the directory realm/domain from_
    user that is mapped. Useful when mapping
    LDAP identities to POSIX users
    Drops all groups from each object's ACL.
    with --acl-add-grants to add specific_
    grants instead
    Drops all users from each object's ACL._
    with --acl-add-grants to add specific_
    grants instead

Decryption Filter (decrypt)
    Decrypts object data using the Atmos Java SDK encryption standard
    (https://community.emc.com/docs/DOC-34465). This method uses envelope encryption_
↳ where
    each object has its own symmetric key that is itself encrypted using the master
    asymmetric key. As such, there are additional metadata fields added to the object_
↳ that
    are required for decrypting
    --decrypt-keystore <keystore-file>
↳ that
    required. the .jks keystore file_
    holds the decryption keys. which_
↳ key to
    use is actually stored in the object
    metadata
    the keystore password
    by default, the modification time
    (mtime) of an object does not change
    when decrypted. set this flag to_
    the mtime. useful for in-place
    decryption when objects would not
    otherwise be overwritten due to_
    timestamps
    by default, if an object is not
    encrypted, it will be passed_
    filter chain untouched. set this_
    fail the object if it is not_
    --fail-if-not-encrypted
↳ through the
↳ flag to
↳ encrypted

```



**Encryption Filter (encrypt)**

Encrypts object data using the Atmos Java SDK encryption standard (<https://community.emc.com/docs/DOC-34465>). This method uses envelope encryption.

↪ where each object has its own symmetric key that is itself encrypted using the master asymmetric key. As such, there are additional metadata fields added to the object.

↪ that are required for decrypting. Note that currently, metadata is not encrypted.

↪ --encrypt-force-strong 256-bit cipher strength is always used

↪ if available. this option will stop operations if strong ciphers are not available

↪ --encrypt-key-alias <encrypt-key-alias> the alias of the master encryption key

↪ key within the keystore

↪ --encrypt-keystore <keystore-file> the .jks keystore file that holds the

↪ the master encryption key

↪ --encrypt-keystore-pass <keystore-password> the keystore password

↪ --encrypt-update-mtime by default, the modification time (mtime) of an object does not change when encrypted. set this flag to update

↪ the mtime. useful for in-place encryption when objects would not otherwise be overwritten due to matching

↪ timestamps

↪ --fail-if-encrypted by default, if an object is already encrypted using this method, it will be

↪ passed through the filter chain untouched. set this flag to fail the object if it is already encrypted

**Gladinet Mapper (gladinet-mapping)**

This plugin creates the appropriate metadata in Atmos to upload data in a fashion compatible with Gladinet's Cloud Desktop software when it's hosted by EMC Atmos

↪ --gladinet-dir <base-directory> Sets the base directory in Gladinet to load content into. This directory must already exist

**ID Logging Filter (id-logging)**

Logs the input and output Object IDs to a file. These IDs are specific to the

↪ source and target plugins

↪ --id-log-file <path-to-file> The path to the file to log IDs to

**Local Cache (local-cache)**

Writes each object to a local cache directory before writing to the target.

↪ Useful for applying external transformations or for transforming objects in-place (source/target are the same)

NOTE: this filter will remove any extended properties from storage plugins (i.e.,

↪ versions,

```
CAS tags, etc.) Do not use this plugin if you are using those features
--local-cache-root <cache-directory> specifies the root directory in which to_
↪cache
                                files

Metadata Filter (metadata)
  Allows adding regular and listable (Atmos only) metadata to each object
  --add-listable-metadata <name=value,name=value,...> Adds listable metadata to_
↪every
                                object
  --add-metadata <name=value,name=value,...> Adds regular metadata to_
↪every
                                object

Override Mimetype (override-mimetype)
  This plugin allows you to override the default mimetype of objects getting
  transferred. It is useful for instances where the mimetype of an object cannot be
  inferred from its extension or is nonstandard (not in Java's mime.types file)._
↪You can
  also use the force option to override the mimetype of all objects
  --force-mimetype If specified, the mimetype will be overwritten
                    regardless of its prior value
  --override-mimetype <mimetype> Specifies the mimetype to use when an object has_
↪no
                                default mimetype

Preserve ACLs (preserve-acl)
  This plugin will preserve source ACL information as user metadata on each object

Preserve File Attributes (preserve-file-attributes)
  This plugin will read and preserve POSIX file attributes as metadata on the object

Restore Preserved ACLs (restore-acl)
  This plugin will read preserved ACLs from user metadata and restore them to each
  object

Restore File Attributes (restore-file-attributes)
  This plugin will restore POSIX file attributes that were previously preserved in
  metadata on the object

Shell Command Filter (shell-command)
  Executes a shell command after each successful transfer. The command will be_
↪given two
  arguments: the source identifier and the target identifier
  --shell-command <path-to-command> The shell command to execute
```

---

### Starting the ecs-sync Service

---

(this page applies to ecs-sync 3.0+)

The **ecs-sync OVA** comes with ecs-sync installed and running as a service. However, if you're **not** using the OVA, you need to start ecs-sync in REST mode so you can submit jobs via XML configuration file. The best way is to install it as a service (the same way the OVA is configured). If that's not an option, you can also manually start the service to run in the background. To do this, simply run the following:

```
nohup java -jar ecs-sync-3.0.jar --rest-only > /var/log/ecs-sync.log &
```

This will start ecs-sync in the background in REST mode detached from the current console (it will still run after you exit the shell). The logs will go to `/var/log/ecs-sync.log` (it's also a good idea to rotate this via logrotated).

**Note:** only one instance of ecs-sync should be running at a time. To be sure you're not running more than one, check for existing instances with ps:

```
ps -ef | grep java | grep ecs-sync
```

### Preparing the Configuration File

Ecs-sync is designed to be run from a submitted xml file that contains all necessary options, addresses, and credentials. This file can be created by hand (there are examples in `ecssync/ecs-sync-[version]/sample`) or easily via the newly included XML Generator. A guide to the XML Generator can be found [here](#). Once a proper xml configuration file has been created and modified with the correct information, you're ready to begin your sync.

**Note:** filters will be applied in the order they are specified in the XML (this is true for legacy-cli and UI as well)

### Starting a Sync

To start a sync, you should run the following:

```
ecs-sync-ctl --submit <config-file>.xml
```

Where `<config-file>.xml` is the path to your configuration XML file. Note that the XML format has drastically changed in 3.0 given the new universal configuration model. There are several sample XML files [here on github](#) or on the OVA in `~ecssync/ecs-sync-3.1/sample/`. Use these as a guide.

The above command will return a job ID. It's important to keep track of this ID so you don't confuse this job with another.

Note that all of the commands on this page assume you are using the OVA, which has a pre-configured path to make these commands easier to run. However, if you are not using the OVA, be aware that you will not have the scripts in your path. The scripts are located in the `ova/bin/` directory of the distribution.

## Alternate (legacy) CLI execution

You can also execute a sync in a separate process by passing CLI arguments directly to the `ecs-sync` jar. Prior to 3.0, this was the standard method of executing most syncs. Note that when running in a separate process, when the sync completes, the REST server dies, so you lose the ability to query status info (you will have to check the log file to see the results of the sync).

To run a sync with an XML configuration via the CLI:

```
nohup java -jar ecs-sync-3.0.jar --xml-config <config-file>.xml > <log-file>.log &
```

You can also pass the entire configuration as CLI parameters instead of using an XML file. Please refer to the full CLI syntax for all available options.

## Checking status

To check status of all syncs, use the `--list-jobs` command like so:

```
ecs-sync-ctl --list-jobs
```

This will list all the jobs the service is aware of and what their status is. It's important to keep track of your job IDs so you can tell them apart.

To list detailed status of a specific job, use the `--status` command:

```
ecs-sync-ctl --status <job-id>
```

Where `<job-id>` is the job ID of the job.

## Changing Thread Count

To change thread count use the `--set-threads` command:

```
ecs-sync-ctl --set-threads <job-id> --threads 32
```

The above will set the thread count to 32 for the `<job-id>` job. Note that changing thread counts happens gracefully, so if you reduce the thread count, running threads are allowed to finish their transfers before being shut down.

## Pausing/Resuming/Stopping

To pause a job:

```
ecs-sync-ctl --pause <job-id>
```

This operation gradually pauses the job by stopping new objects from entering the transfer pool. Existing objects are allowed to finish.

To resume a job:

```
ecs-sync-ctl --resume <job-id>
```

This will resume the processing of new objects in the transfer pool exactly where it left off when a pause operation was executed.

To completely stop and abandon a job:

```
ecs-sync-ctl --stop <job-id>
```

This is behaviorally the same as pause, except that you cannot resume the job.

## Deleting a job

You may notice over time that there are many jobs listed by the service and it may become confusing to sort them all. For this reason, it is recommended that after you are satisfied with the completion of a job and have collected any useful information from it (note the database also contains detailed info as well), you should delete the job from the list. You will no longer be able to see the job summary after this is done.

```
ecs-sync-ctl --delete <job-id>
```

A job must be stopped or completed before deleting it.



---

### Troubleshooting the UI

---

#### Grails Error on First Login

This is expected and can be ignored. The error will resolve once a new user email is successfully submitted to the UI.

#### Attempting to Initialize Shows “Missing Configuration” Error

Sometimes attempting to initialize will show the “Missing Configuration” error. This is likely because the user has entered their email address and pressed [ enter ]. This action will throw the above error every time. The user must **\*click\*** the “Save & Write Configuration to Storage” button for the email to be correctly saved. Until the email is successfully saved the UI will remain un-initialized and the user will not be able to proceed.

#### “Sync Options” fields required when creating a new sync

These fields are not intended to be required. This is a documented bug in 3.1 that will be addressed in 3.1.1. The problem shows up when using Chrome, Internet Explorer, and Firefox browsers. To get around this show **\*all\*** advanced options fields and enter [space] into each of the empty fields. Doing this will allow the user to submit a new sync.

#### Unexplained CAS object failures

If you cannot find an explanation for CAS object failures, try turning on CAS SDK logging, which may provide additional info.





## CHAPTER 10

---

### What is ecs-sync?

---

ecs-sync is a tool designed to migrate large amounts of data in parallel. This data can originate from many different sources.



# CHAPTER 11

---

## Why use ecs-sync?

---

There are many reasons why you may need to migrate data. Maybe your application team is starting to embrace the object paradigm and wants existing files to become objects. Or perhaps you need to move sensitive data out of a public cloud. No matter the reason, ecs-sync can probably help. It was written specifically to move large amounts of data across the network while maintaining app association and metadata. With ecs-sync, you can pull blobs out of a database and move them into an S3 bucket. You can migrate clips from Centera to ECS. You can even zip up an Atmos namespace folder into a local archive. There are many use-cases it supports.



## CHAPTER 12

---

### What it Does

---

Using a set of plug-ins that can speak native protocols (file, S3, Atmos and CAS), ecs-sync queries the source system for objects using CLI or XML-configured parameters. It then streams these objects and their metadata in parallel across the network, transforming/logging them through filters, and writes them to the target system, updating app/DB references on success. There are many configuration parameters that affect how it searches for objects and logs/transforms/updates references. See the CLI Syntax section below for more details on what options are available.