

Active Circle

Ingest Guide

Ingestion of an external filesystem into a share

Ingest Guide: Active Circle Storage System

Version 7.1

Published October 2026

Revision 1.0

Copyright © 2026 Active Circle. All rights reserved.

The software described in this document is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer

language, in any form or by any means — electronic, mechanical, magnetic, optical, chemical, manual, or otherwise — without the prior written

permission of Active Circle. The Information in this document is subject to change without notice.

Active Circle is a registered trademark of Active Circle SA.

All other marks and names mentioned here may be trademarks of their respective companies.

Active Circle S.A.

Active Circle S.A.

26, rue du Faubourg Poissonnière

75010 Paris

FRANCE

Tel: +33 1 34 65 90 29

www.active-circle.com

INDEX OF FEATURES

OVERVIEW.....	4
BASIC CONCEPTS.....	4
USAGE.....	9
MONITORING, SUPERVISION AND AUDIT.....	20
ERROR HANDLING.....	23
LIMITS AND CONSTRAINTS.....	24
LICENSE	25

Overview

The Ingest feature allows you to import into the VFS (Virtual FileSystem) of an Active Circle share, all files located in a filesystem accessible, locally or remotely, from an Active Circle node.

This accessible filesystem together with the access node constitutes a new concept called **External Source**, which will be created in the catalog.

Ingested files are not copied into Active Circle at ingestion time: each ingested file is registered in the share's VFS and associated with a data location addressing the external file, so that Active Circle can process it (storage, read-back, archiving, checksum computation, destaging, and so on) while the data remains on its original storage.

Basic concepts

Principle

Ingest is an asynchronous, near-continuous import: the external filesystem is walked through on demand. For each new file encountered during the walk, an object is created in the VFS of the destination share. These objects, files or directories, have a new kind of data location, a so-called **External Data Location**, which contains the access path of the data on the external filesystem together with a reference to the node where the data is accessible. This data location is used in the same way as the other data locations (cache, stream or archive) to read the file back.

If a file has been deleted on the external filesystem during the walk, it is not deleted from Active Circle, but its **External Data Location** will not be readable anymore and will be blacklisted on the next read of the file. If a file has been modified, a new version can be created, according to a policy describing the behavior to adopt in such a case. These policies are called **Ingest Existing File Policy** (see the Usage section), and the **External Data Location** of the previous version is deleted.

During an Ingest, the disk copies (Stream) of the ingested files can be created, so that the data is protected in the Active Circle secured storage pools; the storage policy is triggered after stream creation.

Ingest is only allowed on plain in service shares, but not on WORM shares.

Ingest Parameters

The Ingest process behaviour is driven by a set of parameters so called **Ingest Parameters**. They define how the external filesystem is walked through (recursivity, name-based filtering), how the ingested objects are created in the VFS (owner preservation, import of the root of the ingested filesystem), the **Existing File Policy**, i.e. how the existing files of the share are handled when an ingested file already exists (skip, overwrite, versioning, error), and whether the disk copies (Stream) of the ingested files are created. The parameters also define the criteria used to detect that a file has changed since a previous ingestion (date and size, or content checksum).

External Source

An **External Source** is the combination of:

- an Active Circle node providing access to the external filesystem (the access node);
- the path of the external filesystem on that node (the mount point).
- **Ingest Parameters** to use when an Ingest imports data from this **External Source**

External Sources are declared catalog objects, managed in the administration interface: they are visible in the **Storage** view under the new **Externals** node (cf Fig 1).

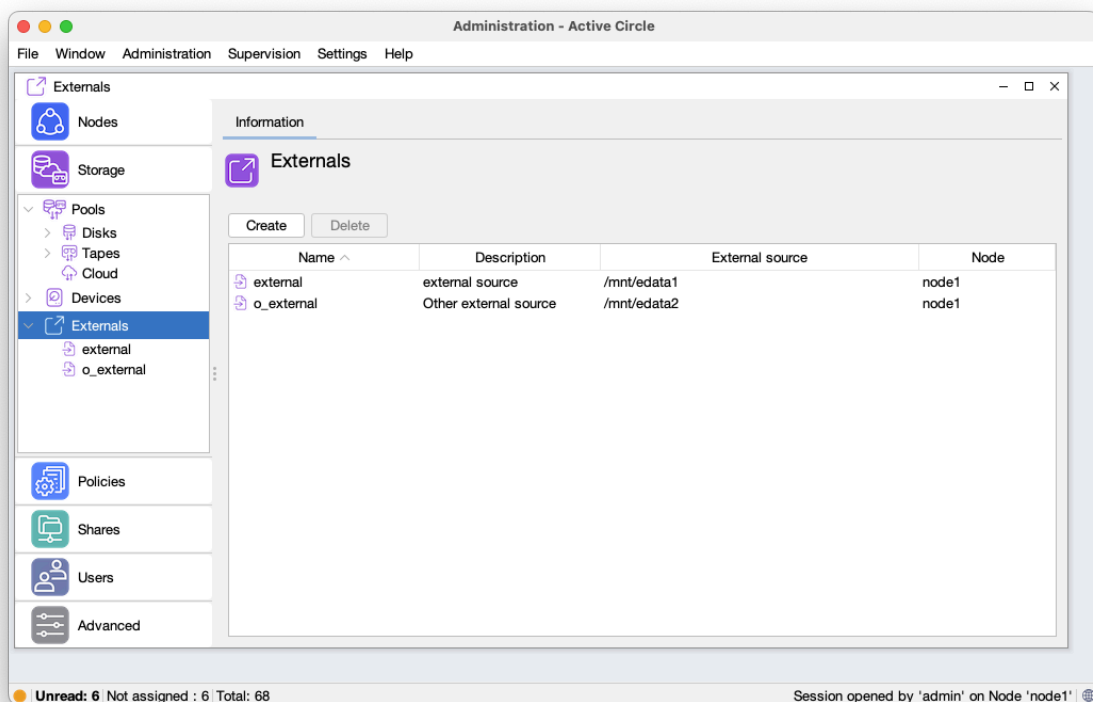


Fig 1. External Sources in the administration interface (Storage view)

External Data Location

During the ingestion, a new entry is created in the catalog for each new object encountered. These entries contain a new data location to address the external data called **External Data Location**.

In any case an **External Data Location** gives the node where the data can be accessed and its path.

Internally, the path can be either implicit, i.e. calculated from its parent **External Data Location** and the relative name of the file itself, or explicit, i.e. stored in the **External Data Location**.

The **External Data Location** can have a direct description of the external volume: a reference to the access node together with the absolute path of the source filesystem on that node (mount point). This is the form used when the Ingest is directly given the node and the mount point.

It can also have an indirect description of the external volume: a reference to a declared **External Source**. In this case, explicit **External Data Location** contains the relative path to the mount point of the external object.

Indirect form of the **External Data Location** allows to keep the access of the external data if the mount point or the access node has changed by editing the corresponding **External Source**.

In addition to the access information, the **External Data Location** records the size and the modification date (mtime) of the ingested file, and if required by the **Ingest Parameters** the CRC64 checksum of the file. These values, as recorded at ingestion time, are used at read-back to verify that the file has not changed on the external source since the ingestion (see Reading from an External Data Location).

The **External Data Location** is visible in the file location information: an **External** node has been added to the upper part of the data location frame in the explorer, showing the name of the server that provides access to the external source and the path in the filesystem tree of the external source (cf Fig 2).

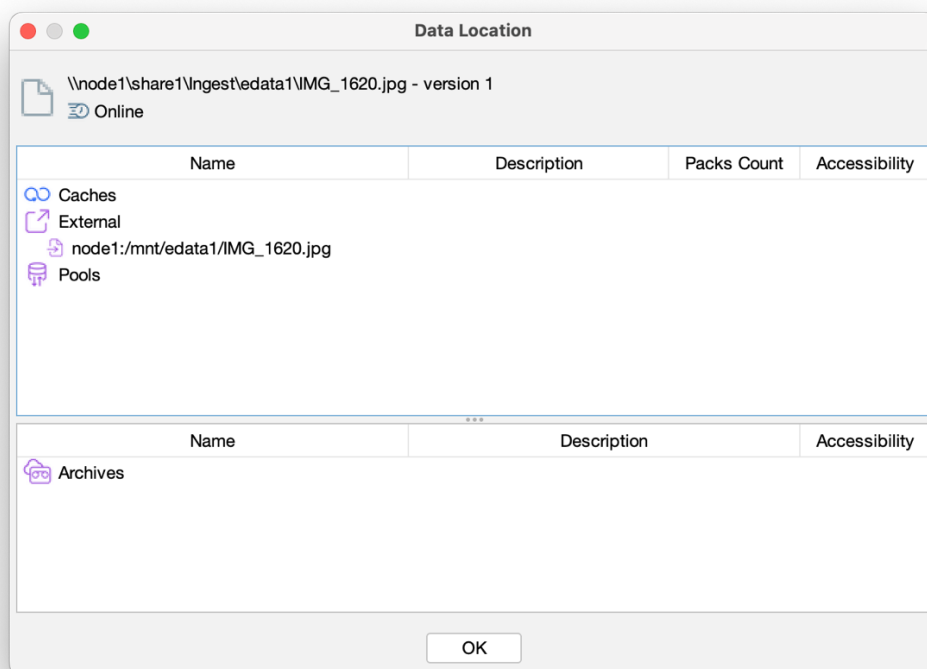


Fig 2. External Data Location in the explorer interface

Versioning

When a file that already exists in the share is encountered during an Ingest, the Existing File Policy decides of its fate (see **Ingest parameters**): the file can be skipped (SKIP), the Ingest aborted (ERROR), or be ingested again (OVERWRITE or VERSIONING). The following applies to the two latter policies, when the file has changed since the last ingestion; if the file has not changed (same size, modification date and, when computed, same checksum), no new version is created and the file is simply reported as unchanged, whatever the policy.

When the file is ingested again, the ingested file becomes the new current version of the file in the VFS:

- with the **VERSIONING** policy, the versioning rules of the share apply: on a historized share, a new version is created and appended to the file history — except when the previous version has no copy in the secured storage pools, in which case the previous version is simply overwritten; on a non-historized share, the current version is always overwritten;
- with the **OVERWRITE** policy, the previous version is always overwritten.

In one case, however, the previous version is never overwritten, whatever the policy and the share: if it has a copy in the secured storage pools — a disk copy (Stream), even being created, or an archived copy — a new version is created instead. Only a version whose data locations are the external source or the cache, i.e. with no durable copy, can be overwritten.

The new version gets the **External Data Location** of the ingested file. The external data locations are removed from all the previous versions of the file, and a previous version is entirely deleted when, after this removal, it has no data location left — i.e. it had no stream nor archive copy and was only referenced through the **External Data Location**. As a consequence, only the versions with a storage in the secured pools are kept in the history.

Reading from an External Data Location

When the best data source to read a file is its **External Data Location**, Active Circle reads through the access node, at the system path given by the **External Data Location**.

When the source is remote, the reading is delegated to the access node that holds the location.

Before using an **External Data Location**, Active Circle checks that it has not been deleted. If such a case occurs, another data source is used.

Active Circle verifies that the file has not changed on the source: if the file size or the modification date differs from the values recorded at ingestion time, the **External Data Location** is blacklisted for this file and another data source is used. When a checksum was computed at ingestion time, a CRC64 checksum is recomputed at read-back and compared with the value stored in the **External Data Location**, in order to detect a change of the file content; if the values differ, the read is considered as failed and the **External Data Location** is blacklisted, so that another available data source is used.

Selecting the data source. When several data locations of a file exist (external data location, cache copies, stream, archives), Active Circle selects the best available data source, in the following order:

1. Local **External Data Location**.
2. If the circle parameter `activecircle.nas.readFirstFromSameDomainCell` is set to `true`: the cache of the nodes of the same domain, then an **External Data Location** of the same domain.
3. Disk copy (Stream).
4. The cache of the remote nodes, then the remote **External Data Location** — if they have not been tried in the first pass, or if the circle parameter `activecircle.nas.readFromOtherDomainCellBeforeFromArchive` set to `true`. In this latter case, the search of the remote caches and of the **External Data Locations** is not limited to the domain.
5. Archives.
6. **External Data Location**.
7. Remote cache. This case must be rare, since it corresponds to files that are not protected yet.

The **External Data Locations** are therefore tried as early as possible in the selection, and the remote cache remains the last resort, since it is not a durable copy.

Usage

Launching an Ingest

In this version, the ingestion is always launched on demand, either from the AC explorer or from the command line. It can be launched by any user having full access to the share (Browse, Create and Write); the data administration profile is not necessary.

An ingestion is performed once. However, nothing prevents from launching an ingestion several times for the same **External Source**, with the same parameters or not.

From AC explorer

Navigate in the VFS of the share and position on a directory, or on the root of the share — which defines the destination directory of the ingestion — in the context of a node. The **Ingest** menu item of the context menu (cf Fig 3) opens a dialog to enter the ingest parameters:

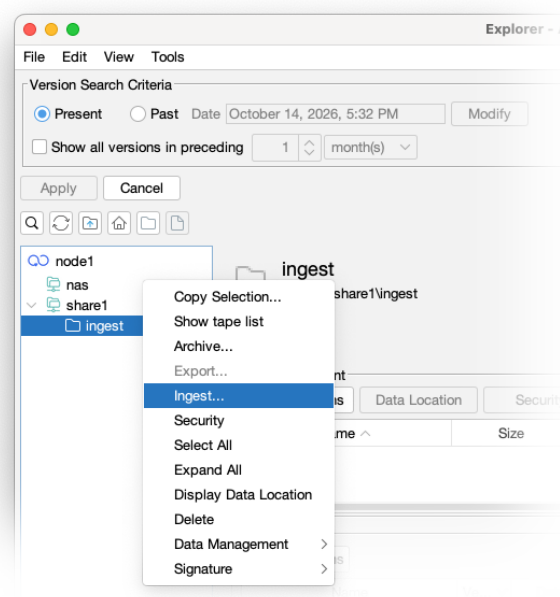


Fig 3. Ingest menu item

In the ingest dialog you must choose (cf Fig 4):

- the external data source, either chosen among the **External Sources** accessible from the current node, or defined manually as a mount point together with the access node (pre-filled, since it is contextual);
- the destination directory (pre-filled, since it is contextual);
- the **Ingest Parameters** (cf Ingest parameters).

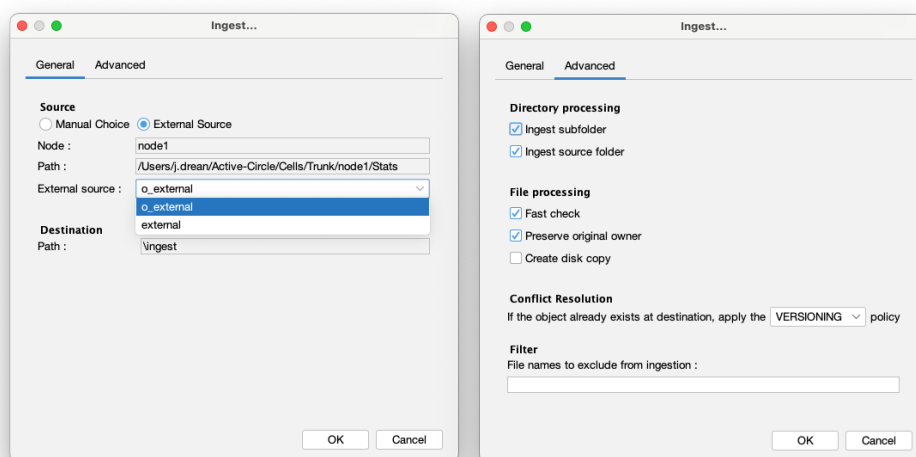


Fig 4. Ingest dialog : (a) General tab - (b) Advanced tab

Ingest parameters

The behaviour of an ingestion is driven by a set of so-called **Ingest Parameters**. They define how the external filesystem is walked through (recursivity, name-based exclusions), how the ingested objects are created in the VFS (owner preservation), the **Existing File Policy** — i.e. how the files of the share are handled when an ingested file already exists —, the criteria used to detect that a file has changed since a previous ingestion, and whether the disk copies (Stream) of the ingested files are created.

When the ingestion is started from a declared **External Source**, the default values of the **Ingest Parameters** are those defined in this **External Source** (the so-called preferred parameters); otherwise, the defaults are the values given in the table below. In any case, the parameters explicitly given at launch override the defaults.

The table below lists the Ingest Parameters with their default value when no External Source is used.

Parameter	Default	Description
Ingest subfolder	yes	Ingest the complete subtree of the mount point; if disabled, only the first level is ingested.
Ingest source folder	yes	Create the root of the ingested file tree in the destination folder; if disabled, only the content of the mount point is ingested.
Fast check	yes	Define the comparison used to check if a file has changed: if true, only the size and the modification time (mtime) are compared; if false, a CRC64 checksum is computed upon ingestion.
Preserve original owner	yes	Preserve the owner and the group of the ingested objects; if disabled, use the session identity as owner (UID) and group (GID) of the ingested objects.
Create disk copy	no	Create the disk copies of the ingested objects, so that the data is protected in the secured storage pools; the storage policy is triggered after stream creation. Only applies if the share has a storage policy.
Conflict resolution (Ingest Existing File Policy)	VERSIONING	The rule applied when an ingested file already exists in the share, can be: SKIP (ignore the file and keep the previous version), OVERWRITE (replace the previous version by the ingested version), VERSIONING (apply the versioning rules of the share: if it is historized, create a new version with the external data location, otherwise replace the previous version), ERROR (abort the ingestion).
Filter	-	Semicolon-separated list of regular expressions matching the file and/or directory names not to ingest. Names are case-sensitive; prefix with (?i) for case-insensitive. End with / for directories only, or /* for directories and files.

Monitoring an Ingest from AC explorer

When an Ingest is launched, a monitoring window follows the Ingest. Its upper part shows the state of the process (e.g. RUNNING), the current step of the Ingest (e.g. INGEST_FILE) and the file or directory currently being processed, updated as the external filesystem is walked through.

The **Details >>** button enlarges the window and unfolds the detail tabs (cf Fig 5):

Progress: the table of the Ingest events, in real time; each ingested, skipped or failed object is reported with its path (the table rows can be copied with the context menu);

Summary: enabled when the process is terminated, it displays the result of the Ingest — the number of ingested files with the ingested volume and the number of skipped files, the number of ingested directories with the number of skipped directories, and the number of failed file

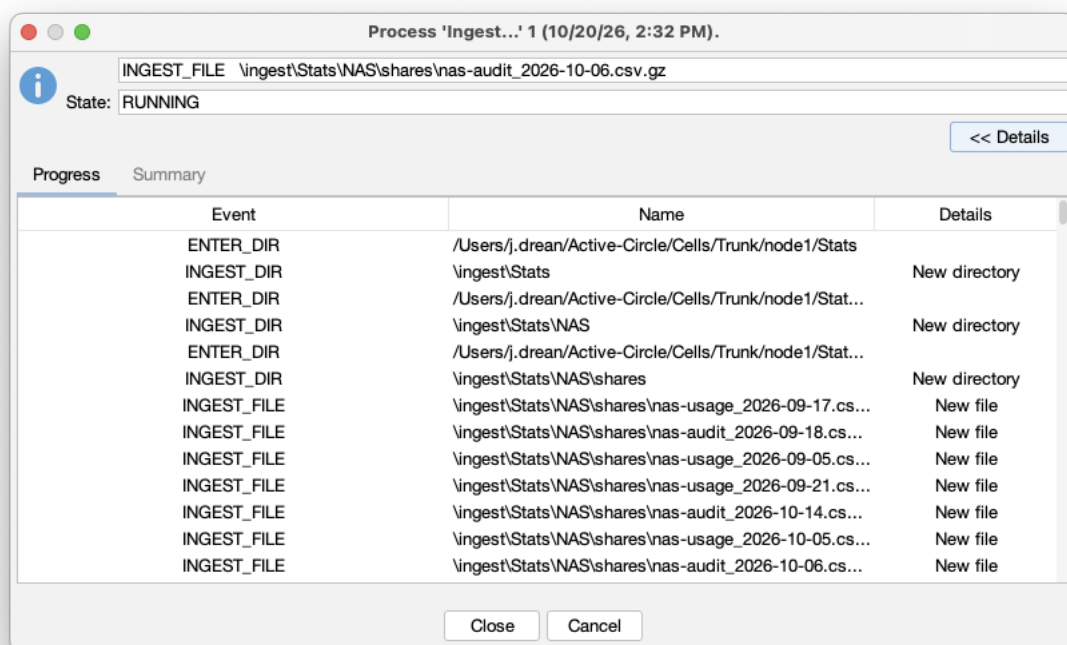


Fig 5. Monitoring a running Ingest

When the process terminates, the termination cause is displayed in the upper part, with an error icon if the Ingest failed, and the summary tab is automatically selected.

As long as the Ingest is running, the **Cancel** button allows to stop the process; the button is hidden when the Ingest is terminated, and only the **Close** button remains to close the window.

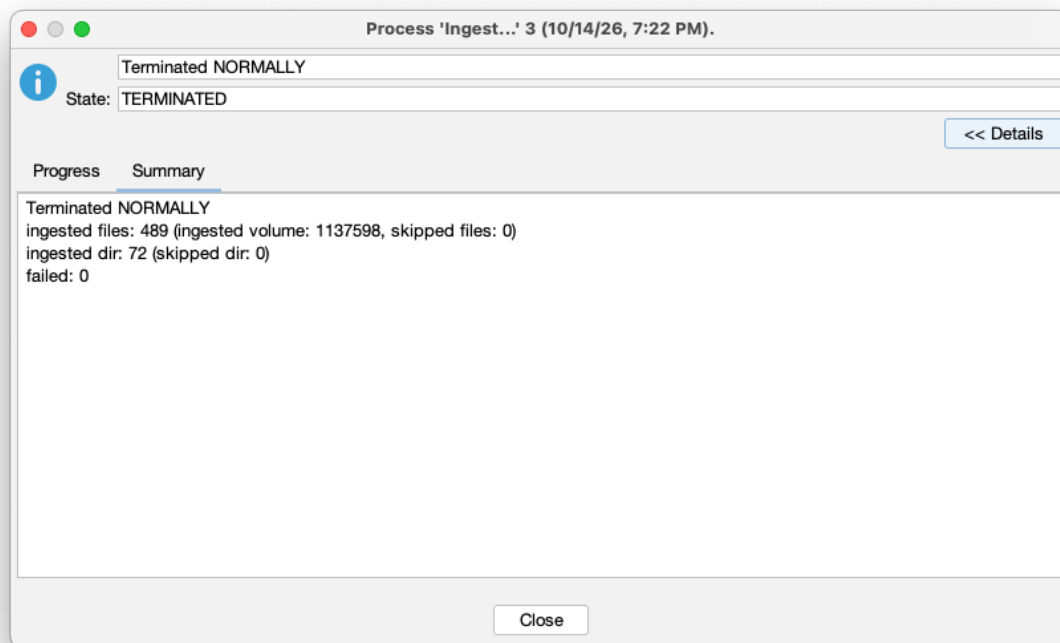


Fig 6. Monitoring a terminated Ingest

From the command line (acigest)

In command mode, the ingestion is launched with the `acigest` command. The destination directory and the source must be specified, as well as the **Ingest Parameters**. The ingestion process is launched asynchronously on the target node.

The main options are the following:

Option	Related Ingest Parameter	Usage
<code>--externalSource</code>	(source)	Name of the External Source to ingest from. Its node must match the processing node. Either <code>--externalSource</code> Or <code>--srcPath</code> must be specified, but not both.
<code>--srcPath</code>	(source)	Absolute path of the source directory to ingest, on the processing node.
<code>--excludeSubFolders</code>	Ingest subfolder (inverted)	Do not ingest recursively the sub-folders of the source directory.
<code>--ingestContent</code>	Ingest source folder (inverted)	Do not ingest the source directory itself, but only its content.
<code>--useSessionId</code>	Preserve original owner (inverted)	Use the session identity as owner (UID) and group (GID) of the ingested objects.
<code>--checksum</code>	Fast check (inverted)	Perform a checksum comparison (CRC64) to detect file changes; if not set, only the size and the last modification time are compared.
<code>--stream</code>	Create disk copy	Create the disk copies of the ingested objects if the share has a storage policy; the option is ignored (with a warning) otherwise.
<code>--existingFilePolicy</code>	Existing File Policy	SKIP, ERROR, OVERWRITE or VERSIONING. Default: VERSIONING.

<code>--excludedFiles</code>	Excluded names	List of excluded file and/or directory names (regular expressions separated by semicolons; \x3B for a semicolon within a name).
<code>--ignorePreferredParams</code>	(preferred parameters)	Do not take into account the preferred Ingest Parameters of the given External Source; without this option, the preferred parameters are used as defaults and every explicitly given parameter override them.

Editing an External Source

External Sources are managed in the administration interface, in the Storage view under the node **External** (cf Fig 1): they can be created, edited or deleted like the other catalog objects.

The edition panel of an **External Source** allows to change its name, its description, its path (the mount point of the external filesystem) and its access node (cf Fig 7), and to define its preferred parameters (cf Fig 8) — the **Ingest Parameters** proposed by default when an Ingest is started from this **External Source**. The preferred parameters are the same as the parameters of the Ingest dialog of the explorer (recursivity, Ingest source folder, Existing File Policy, fast check, owner preservation, disk copy creation, exclusions).

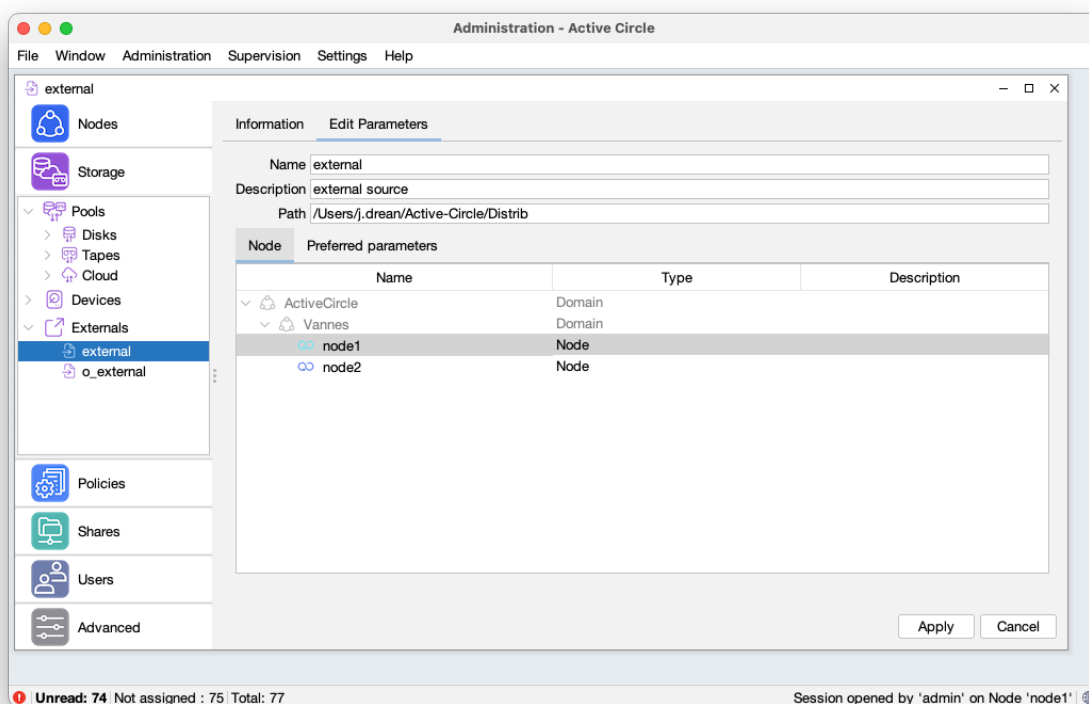


Fig 7. Editing name, description, path and node of an External Source

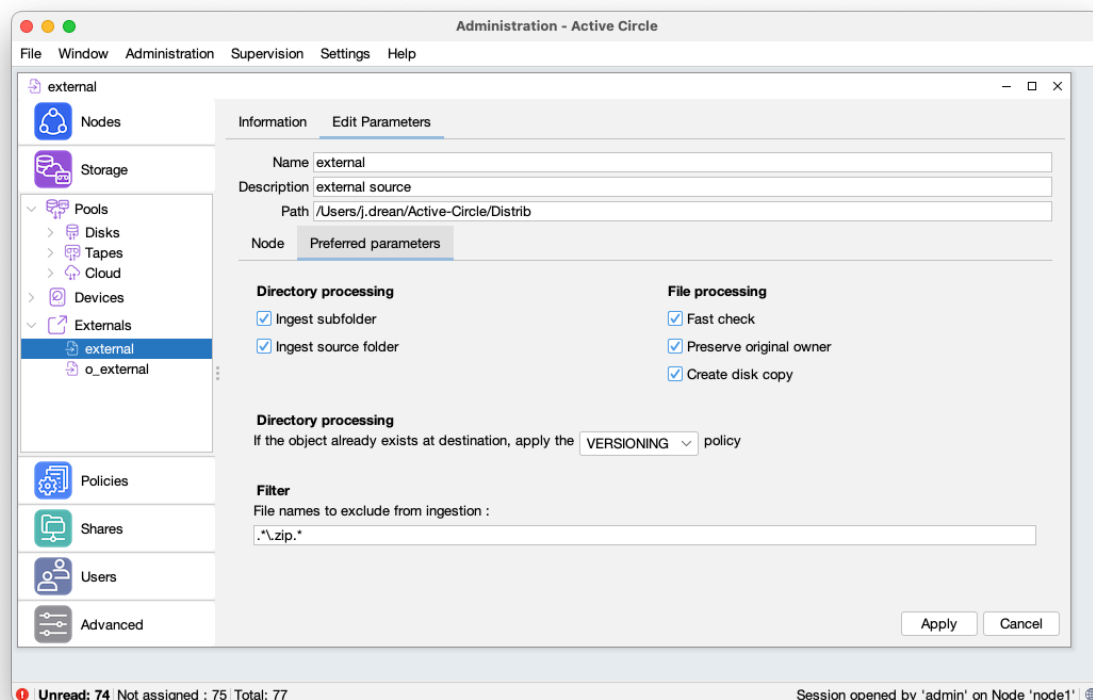


Fig 8. Editing preferred ingest parameters of an External Source

When the node and/or the path of an **External Source** is modified, a confirmation is required: if the new information is incorrect, the access to the files ingested from this **External Source** becomes impossible. Conversely, thanks to the indirect form of the External Data Locations, editing the External Source — for example to replace the access server or to change the mount point — allows to keep the access to the already ingested data without re-ingesting them.



An **External Source** can be deleted only if it is no longer used in the VFS: if some files ingested from this **External Source** still have an **External Data Location** referencing it, the deletion is refused and a message is displayed. To delete such an External Source, the external data locations that reference it must be removed first (see **Removing external copies**), or the concerned files must be deleted.

Removing external copies

The files ingested into a share — whether from a declared **External Source** or directly from a node and a path — are not copied into Active Circle: Active Circle keeps only a link to each file on the external filesystem, its **External Data Location**. Once the data has been secured in the Active Circle storage pools (disk or tape copies created by the storage policy), this link can be removed: it is the prerequisite for deleting an **External Source** still referenced by ingested files (see the Note in Editing an External Source).

Removing External Data Location

Select the files or directories in the AC explorer and launch the removal of the external copies from the context menu (cf Fig 9).

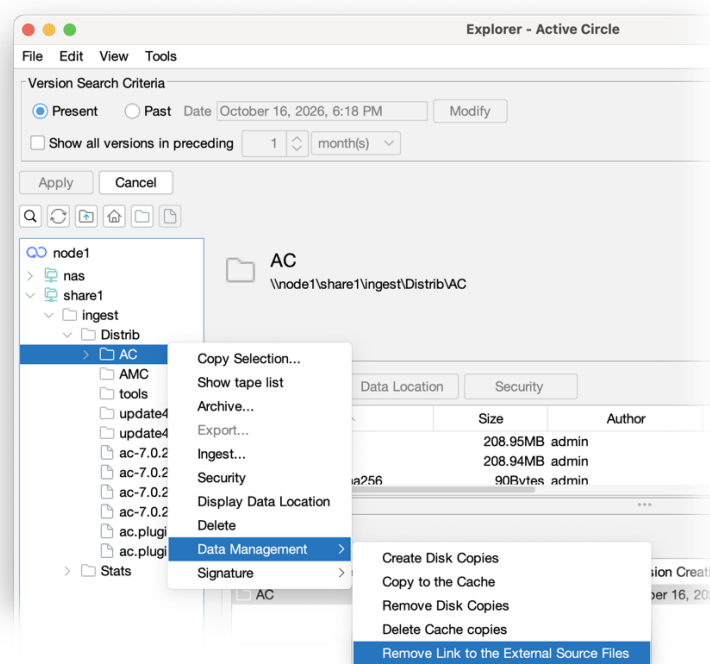


Fig 9. Removing External Data Location menu item

The removal is an asynchronous process: a monitoring window reports its progress and its result (cf Fig 10) — the number of browsed files and the number of removed **External Data Location**. When a directory is selected, the **External Data Location** of all its content are removed; if the removal of some files could not be removed, the directory keeps its own link.

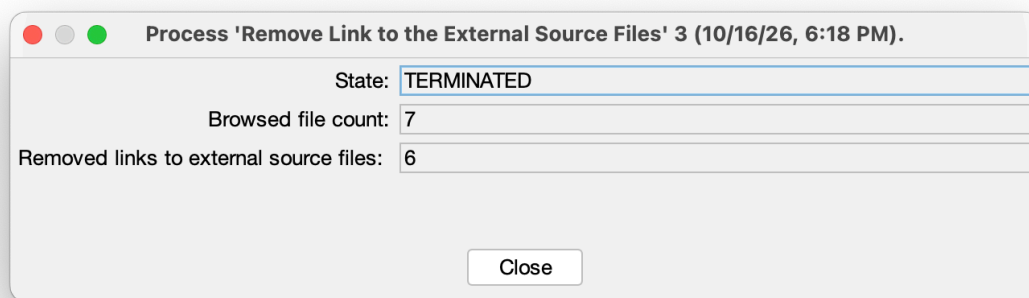


Fig 10. Removing External Data Location result panel

It can also be launched from the command line with `acfileop` and the type `REMOVE_EXTERNAL`, as for the other file operations:

```
acfileop --type REMOVE_EXTERNAL ...
```

Only the external links of secured files are removed: a file is secured when it has a copy in the Active Circle storage pools (disk copy or archive copy). The external link of a file that is not yet secured is kept, without error: the process simply skips it, and the result reports the number of browsed files and the number of removed links — a difference between the two meaning that some files were not secured yet. Once the external link is removed, the file is read from its Active Circle copies, as any other file of the share.

Monitoring, Supervision and audit

Ingest processes

An Ingest is an asynchronous process of the Active Circle job engine: like the other file processes (destaging, unstreaming...), it is visible as an **MJob** of the new **INGESTION** type, displayed in the **Active Job Manager**. The MJob is named after the share and the process number (e.g. `Ingestion:share1#1`), and its description recalls the source of the Ingest (e.g. « Share share1: ingestion process 1 from: /mnt/edata1 »). The list of **INGESTION** jobs gives the node where the Ingest runs, its state (e.g. **SUCCESSFUL**) and its stage (e.g. **TERMINATED**); the dashboard counts the **INGESTION** jobs by state (scheduled, running, unsuccessful, successful) and per node, and the list can be filtered by days, state, node and type (cf Fig 11).

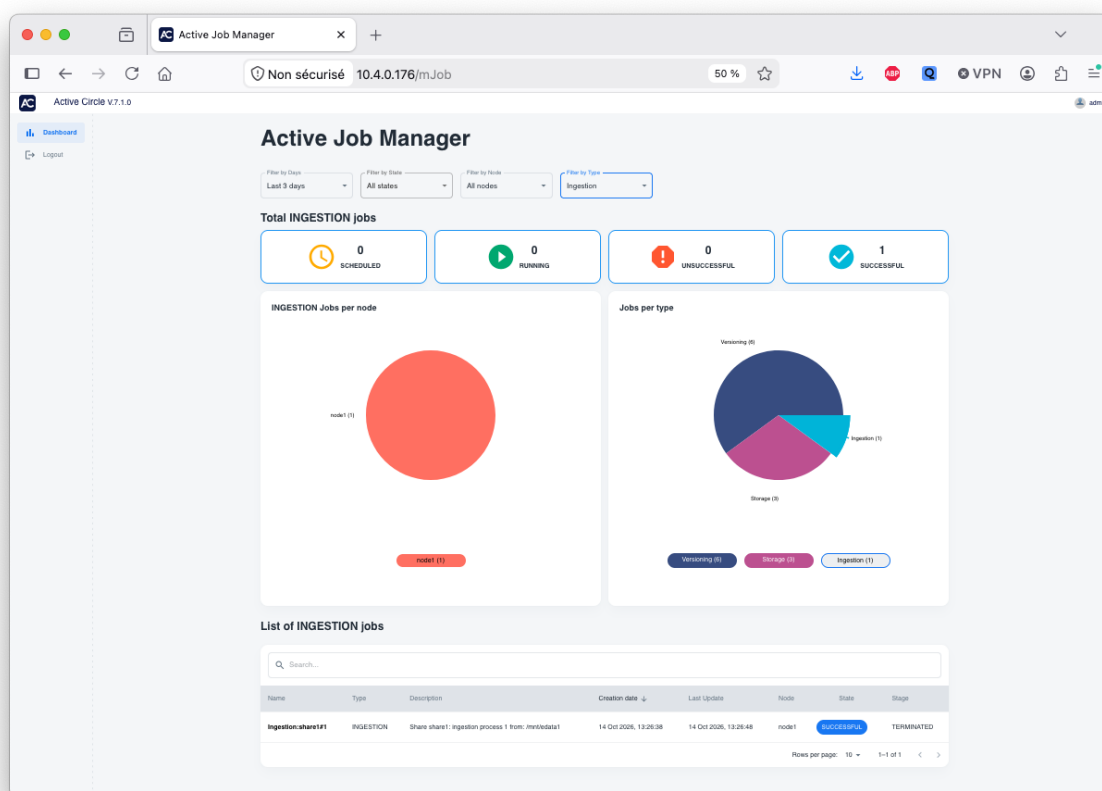


Fig 11. Active Job Manager dashboard showing the INGESTION jobs.

The detail view of the Ingest job (cf Fig 12) gives the result of the Ingest: the number of ingested files and directories, the ingested volume, the number of skipped files and directories, and the number of failed objects — the same figures as the Summary tab of the explorer monitoring window — together with the detail of the ingest events, one entry per object with its path and status (e.g. New file, New directory). The detail can be exported in several formats. The

aggregate counters are updated on each file notification while the Ingest is running; when an Ingest is stopped or fails, the termination cause is reported in the MJob.

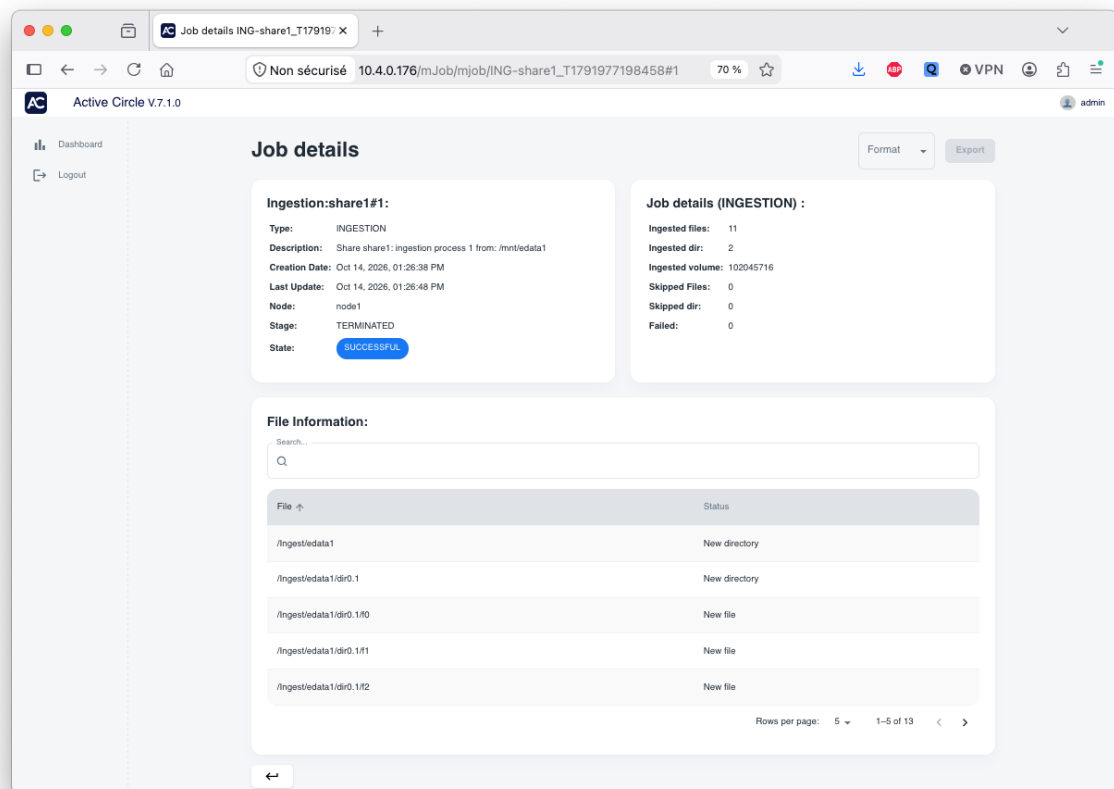


Fig 12. Active Job Manager Ingest job detail.

From the AC explorer, the monitoring window described in the Usage section (see Monitoring an Ingest) shows the same information in real time.

Ingest Events

While an Ingest runs, the ingester emits events that feed the monitoring window of the AC explorer (the Progress table) and the per-file detail of the MJob. The events are the following:

Ingest Event	Raised when
ENTER_DIR	A directory of the source is entered (walk of the tree).
INGEST_FILE	A file is ingested: a new object, or a new version, is created in the VFS.
INGEST_DIR	A directory is ingested: it is created in the VFS at the destination.
SKIP_FILE	A file is skipped — unchanged since a previous Ingest, or not ingested according to the Existing File Policy (SKIP).
SKIP_DIR	A directory is skipped (not ingested according to the parameters).
FAILED	The processing of a file fails: unreadable source, checksum computation failure, and so on.
CREATE_STREAM	The disk copy (Stream) of an ingested file is being created (when the share has a storage policy).

The Progress tab of the explorer monitoring window shows (cf Fig 5), for each event, its type, the path of the object (Name) and the detail message (Details). In the Active Job Manager, the per-file detail of the MJob lists the recorded events with the path of the object (File) and its status (Status, e.g. New file, New directory).

Filtering the recorded Ingest events

The circle parameter `jobManager.recordedIngestEvents` defines which ingest events are recorded as per-file entries in the MJob. Its value is the list of the event categories to record, among: `IngestedFiles`, `IngestedDir`, `SkippedFiles`, `SkippedDir`, `Failed`. By default every category is recorded; an empty value records nothing. The aggregate counters are always updated, regardless of this parameter. The **ENTER_DIR** and **CREATE_STREAM** events have no category and are never recorded as file entries.

Audit

Two audit events are generated during an Ingest: **INGEST_FILE** (label `F.INGES`, disabling mask `0x0001_0000`) for each ingested file, and **INGEST_DIR** (label `D.INGES`, disabling mask `0x0002_0000`) for each ingested directory. The **INGEST_FILE** audit event records the size and the modification date (mtime) of the ingested file as additional values.

Command line

The `ac-ingest` command allows to follow and manage the Ingest processes:

`--show` lists the file processes,

`--showResult` displays them with their results (ingested files and directories, volume, skipped, failed),

`-m/--monitor` waits until the process stops (with `-v 2`, displays the Ingest events)

`--stop/--cancel/--remove` stops, cancels or removes a process (with `-p` to designate it by id).

Error handling

Failures during the Ingest

When a file cannot be ingested — the external filesystem cannot be read, the checksum computation fails, the destination quota is reached, and so on — the failure is reported in the Ingest result (Failed counter) and the Ingest continues with the remaining objects: a file failure does not stop the Ingest. Each failed file is listed in the monitoring window and recorded as a Failed event in the MJob, with its path.

Ingest termination

An Ingest terminates normally when the whole source has been walked through; the result (ingested, skipped and failed counters, ingested volume) is then available in the monitoring window and in the MJob. When an Ingest is stopped or fails, the termination cause is reported:

the state of the process and the cause are displayed in the monitoring window and recorded in the MJob (see **Monitoring, Supervision and audit**).

Ingest level errors

Some errors abort the Ingest itself, before or during the walk:

- the Ingest parameters are invalid (unknown **External Source**, invalid source path);
- the source is not readable from the processing node, or the node of the **External Source** does not match the processing node;
- the **Existing File Policy** is ERROR and an ingested file already exists in the share: the Ingest aborts in error;
- the license does not allow the Ingest feature.

When the Ingest aborts, the objects already ingested before the error remain in the share.

Consistency safeguards

The destination folders are locked during the Ingest: a folder cannot be ingested simultaneously from two different sources, and concurrent modifications of the destination are prevented while the Ingest walks through the tree. The number of files of one Ingest is bounded by the circle parameter `vfs.ingest.maxFiles` (see **Limits and constraints**); when the limit is reached, the Ingest stops in error.

Limits and constraints

In this version, Ingest is on demand only and launched explicitly from the AC explorer or from the command line.

The number of files of an Ingest is limited by the circle parameter `vfs.ingest.maxFiles` defining the maximum number of files of an ingestion process (default 1,000,000 files).

When an **External Source** is used, its node must match the processing node of the ingestion.

The excluded file names are defined by regular expressions and must be chosen carefully to preserve the ingestion performance. The checksum comparison (CRC64) is more accurate than the date and size comparison but requires reading the whole content of the files.

The Ingest feature does not copy the data into Active Circle at ingestion time: the data remains on the external storage, and its availability depends on the external filesystem and on the access node. It is protected against loss only once the stream or archive copies have been created (see Removing external copies in the Usage section).

License

The Ingest feature is subject to a dedicated option in the Active Circle license, granted by the license server as **External ingestion**. The option defines the maximum number of **External Sources** that can be declared in the catalog:

Ingest option	External sources
Level 1	1 external source.
Level 2	Up to 3 external sources.
Level 3	Unlimited number of external sources.

Without the option, the Ingest feature is disabled: creating an **External Source** and launching an Ingest are refused, from the AC explorer, from the command line and from the API.