ZCP 7.0 (build 41322)

Zarafa Collaboration Platform

Zarafa Archiver Manual

ZCP 7.0 (build 41322) Zarafa Collaboration Platform
Zarafa Archiver Manual
Edition 1.0

The Zarafa Archiver provides an integrated archiving solution for Zarafa installations.

---

[1] http://creativecommons.org/licenses/by-sa/3.0/

Introduction

The Zarafa Archiver product provides a way to minimize mailbox sizes by moving older messages to *slower* and thus *cheaper* storage. The slow storage consists of one or more additional Zarafa servers who's sole task is to store archived messages.

The archive Zarafa servers have exactly the same storage architecture as a normal Zarafa server, all MAPI properties are stored in a MySQL database and all attachments are stored compressed on disk.

An archive mailbox can be enabled per user and the user will automatically get access to the archive mailbox from the clients.

Once a message is archived, it can be deleted from the original store. Optionally a stub to the archive can be created that allows a user to see the archived message and open it as if it were a normal message.

The Zarafa Archiver uses the Zarafa's multi-server technology to access archive stores in a seamless way. Nonetheless, the Zarafa Archiver can be used in a single-server setup with limited functionality.

Zarafa Archiver is an additional product and is not a default component of the Zarafa Collaboration Platform. Subscriptions of Zarafa Archiver can only be used with the Zarafa Professional or Enterprise edition, where both editions provide 20 archive users for free.

Conventions

Before starting to install and deploy the Zarafa Archiver it's strongly advised to read this chapter to understand the different terminology.

- **Primary Server**

  The primary server is the server with the best performance and best IO subsystem, that contains the stores on which users normally work.

  > **Note**
  >
  > Although the term **Primary Server** suggests that there's only one primary server, multiple primary servers can exist in a multi-server environment. In this document no distinction will be made between a single-server or multi-server environment unless explicitly stated.

- **Archive Server**

  The archive server is the server, with normally a slower IO subsystem, that contains the archives for the stores that reside on the primary server.

  > **Note**
  >
  > An archive server is just another zarafa-server with the sole purpose of providing storage for one or more archive stores. In a multi-server environment this server will be just another node in the cluster.

  > **Note**
  >
  > Unlike primary servers, there's no need for a multi-server environment to have a multi-archive server setup.

- **Primary Store**

  The primary store is the store that resides on the primary server and on which a user normally works.

- **Archive Store**

  The archive store is the store that resides on the archive server and which is used for storing the archived messages from the primary store.

- **zarafa-archiver**

  The archiver is the application to manage the archiving. Basically it can be used to attach primary stores to archive stores and execute archive runs. It can be installed on any Zarafa server to connect to the primary or archive server using SSL authentication. It can also be used on a single server, using `zarafa-server`'s unix socket.

3

- **Stubbed Message**

  A stubbed message is a message in the primary store that acts as a placeholder for the archived message. These messages occupy virtually no space, but allow a user to see that a message was once there. On top of that it acts as an entry point to the archived version of that message.

- **Archive Configuration**

  An archive store can be configured in two ways, one-to-one and one-to-many. This is not a system wide configuration and can be setup for each archive independently.

  This allows for hybrid systems where N users with small to medium stores can be placed on M archive stores (where M is significantly smaller than N) and users with big to huge stores can be placed on dedicated archive stores.

- **One-to-One Configuration**

  In a one-to-one configuration one archive store is attached to one primary store.

  The advantage of this configuration is that it's faster as the archive store itself is kept smaller.

  The disadvantage is that for each user an additional non-active user needs to be created (since there's a one-to-one mapping between stores and users in Zarafa).

- **One-to-Many Configuration**

  In a one-to-many configuration one archive store is attached to multiple primary stores. For each attached primary store a folder is created in the archive store that will act as the root of the archive for that particular primary store.

  The advantage of this configuration is that less additional non-active users are required.

  The disadvantage is that the archive will become slower if the total amount of archive data in it grows.

- **Single Instances**

  The Zarafa Collaboration Platform uses single instance storage whenever possible to minimize storage requirements when data is stored more than once. The Zarafa Archiver makes use of this technology by remembering which instances it copied to an archive server and referencing that instance whenever possible.

Methods of archiving

The Zarafa Archiver can be implemented for two specific purposes:

1. Archiving for history preservation

2. Archiving for storage optimization

3.1. Archiving for history preservation

In this situation the goal is to archive automatically all incoming and outgoing messages and to keep track of the changes made to the messages.

This implies that for a configurable amount of time each message exists at least two times in the system; once in the primary store and once in the archive store. The used storage can be reduced by stubbing the messages in the primary store.

In this configuration the **`zarafa-dagent`** and **`zarafa-spooler`** are also involved to make sure all incoming and outgoing messages get archived.

3.2. Archiving for storage optimization

In this situation the goal is to minimize the storage capacity requirement for the primary node. A smaller primary database will result in lower server specs, like required RAM. A smaller primary database will also result in faster disaster recovery backups.

To get a smaller database only the most recent messages are kept on the primary node and everything else gets moved to the archive. As a convenience all or some of the messages in the primary store that are copied to the archive node can be stubbed.

This way the capacity requirements will be reduced while still allowing the user to open and search for messages from his primary store.

Installing

4.1. Requirements

To deploy the Zarafa Archiver at least servers are required: one primary Zarafa server and one Zarafa Archive server. When a Zarafa Archiver subscription is available these servers can be configured in a multi-server setup. The multi-server technology is used to connect archives automatically to users.

> **Note**
>
> A multi-server setup requires a central LDAP or Active Directory. It's not possible to use multi-server with the **DB** or **unix** user plugin.

4.1.1. Software Requirements

- ZCP 7.0.4+

- Multi-server Zarafa setup with LDAP or ADS

- Valid ZCP Professional or Enterprise subscription

- Zarafa Archiver subscription

4.1.2. Required Services

The following Zarafa packages are required to install on a primary node:

- zarafa-client

- zarafa-server

- zarafa-licensed (with Zarafa Archiver subscription)

- zarafa-spooler

- zarafa-dagent (for LMTP only)

- zarafa-multiserver

- zarafa-search or zarafa-indexer

- zarafa-utils

The following Zarafa services are required on an archive node:

- zarafa-client

- zarafa-server

- zarafa-utils

- zarafa-multiserver

- zarafa-licensed (with Zarafa Archiver subscription)

4.2. Installation

The Zarafa Archiver software can found on the Zarafa Portal as additional download. The Zarafa Archiver package will include the following two packages:

- zarafa-archiver containing the Archive controller and configuration files

- zarafa-archiver-extra containing additional utilities for archiving setups:

  See *Chapter 11, Zarafa Archiver Extras* for more details.

4.2.1. Default installation location

The zarafa-archiver packages can be installed on any node in the multi-server setup. However, it's recommended to install the packages on the archive node.

This makes sure that the **zarafa-archiver** always has a fast local connection to the archive server.

4.2.1.1. RPM based distributions

Use the following command to install the **zarafa-archiver** and **zarafa-archiver-extra** package on RPM based distributions:

```
rpm -Uvh zarafa-archiver_<version>_<platform>.rpm zarafa-archiver-
extra_<version>_<platform>.rpm
```

Replace **<version>** with the correct version and **<platform>** with the required target platform (**i386**, **i586**, **ia64**, **x86_64**).

4.2.1.2. DEB based distributions

On Debian based distributions use:

```
dpkg -i zarafa-archiver_<version>_<platform>.deb zarafa-archiver-
extra_<version>_<platform>.deb
```

Replace **<version>** with the correct version and **<platform>** with the required target platform (**i386**, **ia64**, **x86_64**).

4.2.2. Activate archiver subscription

To use the archiver subscription on every node archiver subscription has to be placed in the **/etc/zarafa/license** directory of all your servers. Execute the following commands on every node to use the archive subscription:

```
echo 'Archiver code' > /etc/zarafa/license/archiverbase
/etc/init.d/zarafa-licensed restart
```

Additional Archiver CALs can be added in the **/etc/zarafa/license** directory, like normal ZCP CALs.

> **Important**
>
> The archiver subscription must be placed on **all** server nodes, else de-stubbing will not work.

4.3. Basic Configuration

The configuration of the Archive Controller has to be done in **`/etc/zarafa/archiver.cfg`**.

4.3.1. Connection Parameters

As with all Zarafa components, **`zarafa-archiver`** needs to know where to connect to and how to authenticate. This is configured using the **`server_socket`**, **`sslkey_file`** and **`sslkey_pass`** settings.

For instance:

```
server_socket   = file:///var/run/zarafa
sslkey_file     = /etc/zarafa/ssl/certificate.pem
sslkey_pass     = zarafa
```

For more information about creating and configuring SSL certificates, see *http://doc.zarafa.com/7.0/ Administrator_Manual/en-US/html-single/index.html#_creating_ssl_certificates*.

4.3.2. MySQL Settings

`zarafa-archiver` uses one central MySQL database for managing deduplication of archived attachments. The MySQL settings can be configured like this:

```
mysql_host      = localhost
mysql_port      = 3306
mysql_user      = root
mysql_password  =
mysql_socket    =
mysql_database  = zarafa-archiver
```

Configure for storage optimization

For storage optimization there are three important questions:

1. How long do the messages need to exist as normal item in the primary store?

2. How long do the messages need to be accessible from the primary store?

3. How long do the messages need to be accessible from the archive store?

The answers to these questions determine how to configure the **zarafa-archiver**.

5.1. Example 1

Let's assume the answers to the earlier listed archiver questions are:

1. 30 days

2. 1 year (365 days)

3. 5 years (1826 days)

Then the following settings should be set in **/etc/zarafa/archiver.cfg**

```
archive_enable  = yes
archive_after   = 30

stub_enable     = yes
stub_after      = 0

delete_enable   = yes
delete_after    = 365

purge_enable    = yes
purge_after     = 1826

cleanup_follow_purge_after   = yes
```

The **archive_enable = yes** setting enables the archive operation, which is essentially the copying from the primary node to the archive node. The **archive_after = 30** setting causes the archive operation to be performed 30 days after the message was delivered.

The **stub_enable = yes** setting enables the stub operation, which is essentially the operation on the primary message where the body and the attachments are removed. The **stub_after = 0** setting causes the message to be stubbed 0 days after delivery but only if the message is archived. This setting could also be set to 30. Effectively this causes the messages on the primary node to be stubbed immediately after the archive operation.

> **Note**
>
> Set **stub_after** to **30** if the archive on delivery feature is enabled, see *Chapter 6, Configure for history preservation* for more information. Setting **stub_after** to **0** in that case will cause all newly delivered messages to be stubbed on the first archive run.

The **delete_enable = yes** setting enables the delete operation. The delete operation deletes messages from the primary node if they are archived. The **delete_after = 365** causes the delete operation to be performed 1 year after delivery time if the message is archived.

The **purge_enable = yes** and **purge_after = 1826** options cause messages to be deleted from the archive 1826 days after they were delivered on the primary node.

The **cleanup_follow_purge_after** is set to yes to make sure a cleanup run won't delete items from the archive because they don't exist in the primary store anymore.

> ⭐ **Important**
>
> When an item is purged, it's no longer available on the mail system. Please be careful by setting this option.

5.2. Example 2

Now let's assume the answers to the earlier listed archiver questions are:

1. 30 days

2. 30 days

3. Forever

Then the following settings should be set in **/etc/zarafa/archiver.cfg**

```
archive_enable  = yes
archive_after   = 30

stub_enable     = no
stub_after      = 0

delete_enable   = yes
delete_after    = 0

purge_enable    = no
purge_after     = 0

cleanup_follow_purge_after  = yes
```

In this case there's no reason to stub the primary messages as they will be deleted anyway. Therefore **stub_enable** is set to **no**. The **stub_after** setting is ignored.

The **delete_enable** setting is still set to yes, but **delete_after** is now set to **0**, causing the messages to be deleted immediately after they're archived.

The **purge_enable = no** makes sure all messages are kept in the archive forever.

5.3. Unread mail

Another consideration that must be made is if unread messages should be stubbed and/or deleted. One could argue that an unread message shouldn't be stubbed because the user is likely to read it soon and needs fast access. On the other hand one could argue that since the user still hasn't read the message during the time it's available on the primary node, chances are the user isn't going to read it soon.

So if the policy would be to stub unread messages but not delete them, the following settings should be set in **/etc/zarafa/archiver.cfg**

```
stub_unread     = yes
```

```
delete_unread    = no
```

5.4. Other settings

The user has be default read-only permissions to his archive mailbox. When using the Zarafa Archiver for storage optimization it's recommended to give the user write permissions to the archive mailbox. By giving write permissions, the user can remove older email from the archive mailbox by him/herself. The default permissions on the archive mailboxes, can be set with the following option:

```
auto_attach_writable    = yes
```

When a user deletes an item from the primary mailbox which is also available in the archive, the item is by default only removed from the primary mailbox. In this case the item can also be opened again in the the archive mailbox. In an archiving setup for storage optimizations it's recommended to directly delete the item also from the archive mailbox. This can be done, by setting the following option to **delete**.

```
cleanup_action          = delete
```

Configure for history preservation

For history preservation multiple Zarafa components need to be configured.

6.1. Zarafa Dagent

/etc/zarafa/dagent.cfg:

```
archive_on_delivery = yes
```

This causes **zarafa-dagent** to archive each message the moment it is delivered if the store in which the message is delivered is attached to an archive. If the archive operation fails **zarafa-dagent** won't deliver the message at all and return a temporary failure to the MTA.

6.2. Zarafa Spooler

/etc/zarafa/spooler.cfg:

```
archive_on_send = yes
```

This causes **zarafa-spooler** to archive each message it sends if the primary store from which the message is send is attached to an archive. If the archive operation fails **zarafa-spooler** won't send the message and creates a bounce message for the sender.

Sent messages are archived in a special folder in the users archive: *Zarafa Archive/Outgoing*. These messages are not coupled to the copies of the sent messages that can be found in the user's Sent Items folder. This implies that sent messages are archived twice.

6.3. Zarafa Archiver configuration

/etc/zarafa/archiver.cfg:

```
archive_enable        = yes
archive_after         = 0
track_history         = yes
cleanup_action        = store
enable_auto_attach    = yes
```

Setting **archive_enable = yes** and **archive_after = 0** causes the archive to archive every new message and every change on each run of the archiver.

The **track_history = yes** setting causes the archiver to not update the archived copies of messages that were altered after they were archived but to make a new archived copy and move the *old* copy to another special folder in the users archive: *Zarafa Archive/History*.

The **cleanup_action = store** setting changes the behavior of the cleanup operation. Instead of deleting archived copies from the archive when the primary message they reference is deleted, **zarafa-archiver** will move the archived copy to yet another special folder: *Zarafa Archive/Deleted*.

The **auto_attach_writable = no** setting makes sure that the users don't get write privileges on their archives. If they would have write privileges they could alter or delete messages from their archive, making the history useless.

6.4. Optional settings

Because all messages are archived in this setup, it's a possibility to reduce the storage capacity requirements as a bonus.

The following settings could be set in **/etc/zarafa/archiver.cfg**:

```
stub_enable      = yes
stub_after       = 30
stub_unread      = no

delete_enable    = yes
delete_after     = 365
delete_unread    = no

cleanup_follow_purge_after  = yes
```

This causes the archiver to stub all read messages 30 days after they were delivered and delete read messages 1 year after they were delivered. Of course the delete step can also be disabled. The **cleanup_follow_purge_after** is set to yes to make sure a cleanup run won't delete items from the archive because they don't exist in the primary store anymore.

See *Chapter 5, Configure for storage optimization* for more examples.

Archive Management

7.1. Automatic

The **zarafa-archiver** can attach archive stores automatically based on user attributes stored in the LDAP or Active Directory. When using this way of attaching stores, **zarafa-archiver** will create genuine archive stores on the server and attaches the user stores to these archives stores based on the information found in the LDAP or Active Directory.

When using this type attaching and detaching, Outlook and the Webaccess will load the archive stores of the primary stores and all opened shared stores automatically.

To use this feature the **enable_auto_attach** setting must preferably be set to **yes** in **/etc/zarafa/archiver.cfg**:

```
enable_auto_attach = yes
```

Alternatively **zarafa-archiver** can periodically be run to perform the auto-attach operation:

```
zarafa-archiver --auto-attach
```

7.1.1. Using OpenLDAP

To add an archive store to a user through LDAP, the following attribute has to be modified **zarafaUserArchiveStores**. This is is a multi value attribute, which needs to be set to the server name or server names of the servers that should contain an archive store for the user.

A typical ldif could look like this:

```
dn: uid=user,ou=users,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
objectClass: zarafa-user
objectClass: posixAccount
cn: User
gidNumber: 0
homeDirectory: /bin/false
sn: User
uid: user
uidNumber: 1000
givenName: User
mail: user@server.com
userPassword:: e1NTSEF9VzlXV0U3N1NEcW54UkJ3SFJkQUYvVkhrUj
zarafaAccount: 1
zarafaUserServer: userServer
zarafaUserArchiveServers: archiveServer
```

If multiple archives should be attached the ldif might look like this:

```
dn: uid=user,ou=users,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
```

```
objectClass: top
objectClass: zarafa-user
objectClass: posixAccount
cn: User
gidNumber: 0
homeDirectory: /bin/false
sn: User
uid: user
uidNumber: 1000
givenName: User
mail: user@server.com
userPassword:: e1NTEF9VzlXV0U3N1NEcW54UkJ3SFJkQUYvVkhrUj
zarafaAccount: 1
zarafaUserServer: userServer
zarafaUserArchiveServers: archiveServer
zarafaUserArchiveServers: backupServer
```

> **Note**
>
> The archive store won't be created or attached until the next run of **`zarafa-archiver -A`** with **`enable_auto_attach = yes`** or **`zarafa-archiver --auto-attach`**.

7.1.2. Using Active Directory

To add an archive store to a user using ADS, one has to open the user in the *Active Directory Users and Computers* window and select the **`Zarafa Features`** tab.
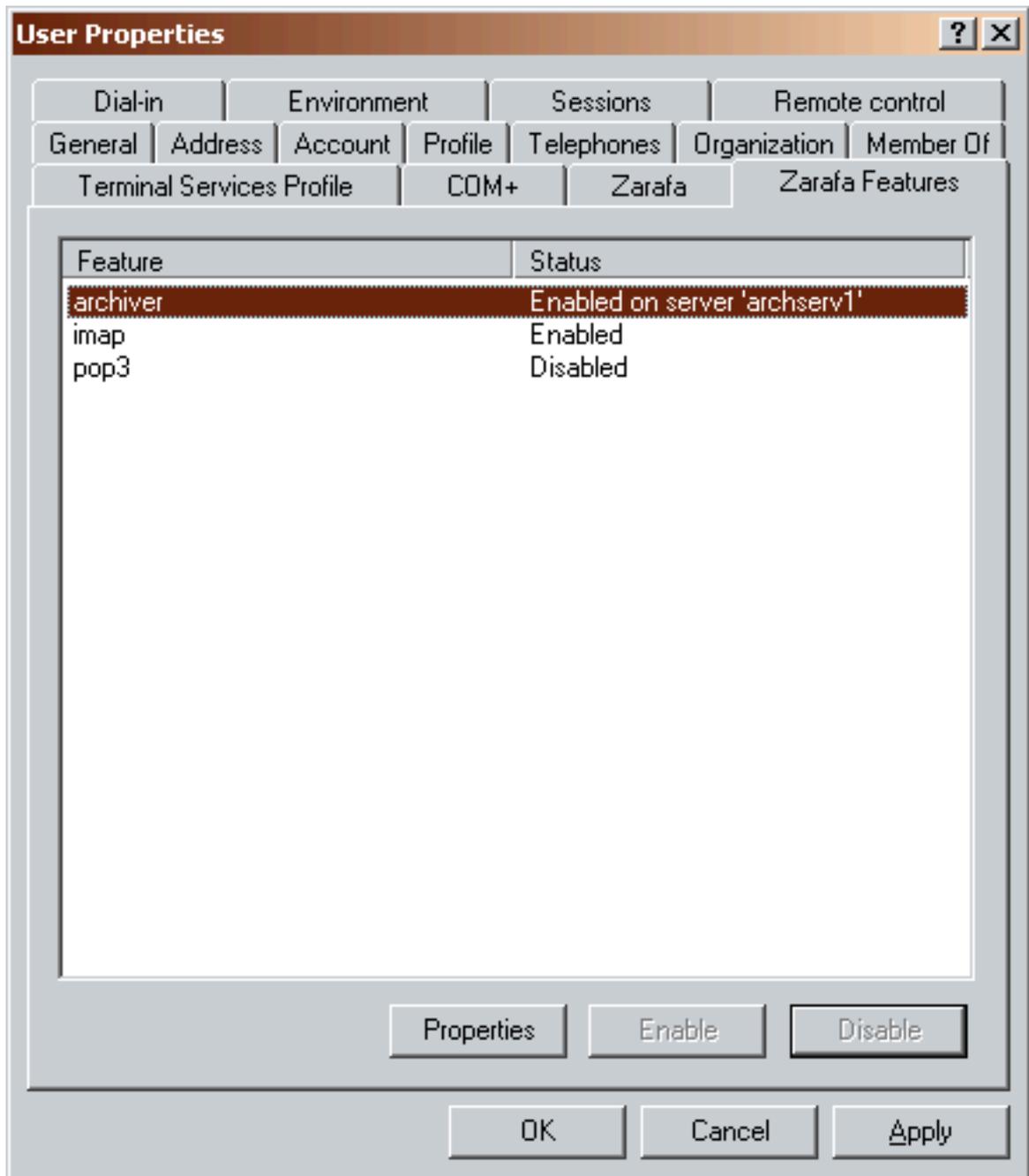
Figure 7.1. ADS Features tab

Next select the **archiver** feature and click **Properties**. This will pop up the dialog in which the server names of the servers on which an archive store should exist for the selected user or users.
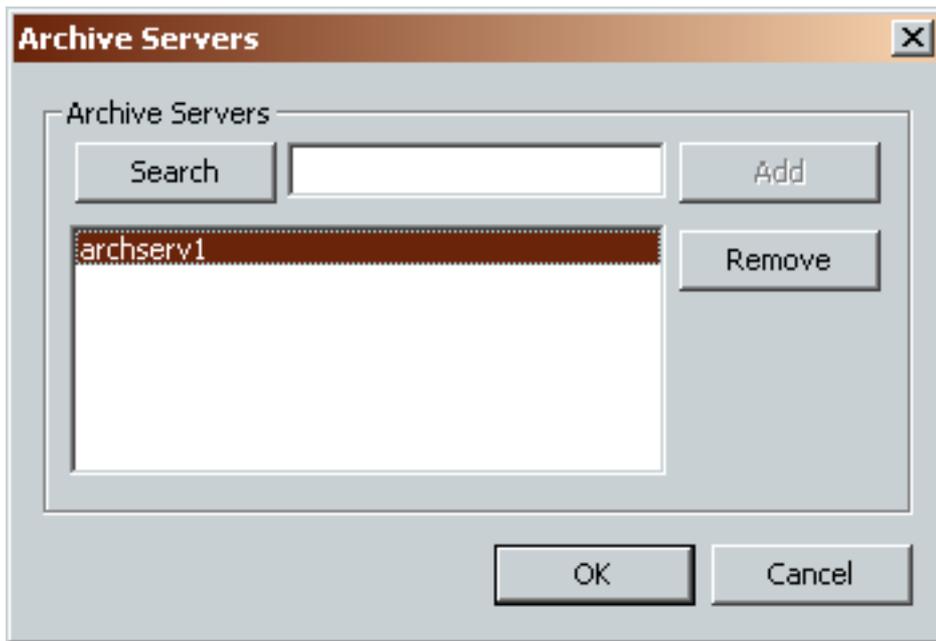
Figure 7.2. ADS Select archive servers

> **Note**
>
> The archive store won't be created or attached until the next run of `zarafa-archiver -A` with `enable_auto_attach = yes` or `zarafa-archiver --auto-attach`.

7.2. Manual

`zarafa-archiver` allows one to attach archives to, detach archives from and list archives of users. A user can have more than one archive attached to allow redundant archives. In those cases these archives are usually on a different server, but that's not mandatory.

7.2.1. Creating an archive store

Creating an archive store is not different from creating a non-active store. So for each archive store a new non-active user must be created in the LDAP/ADS back-end. To keep the address book clean it's recommended (but not required) to hide the user by setting the `zarafaHidden` attribute to 1.
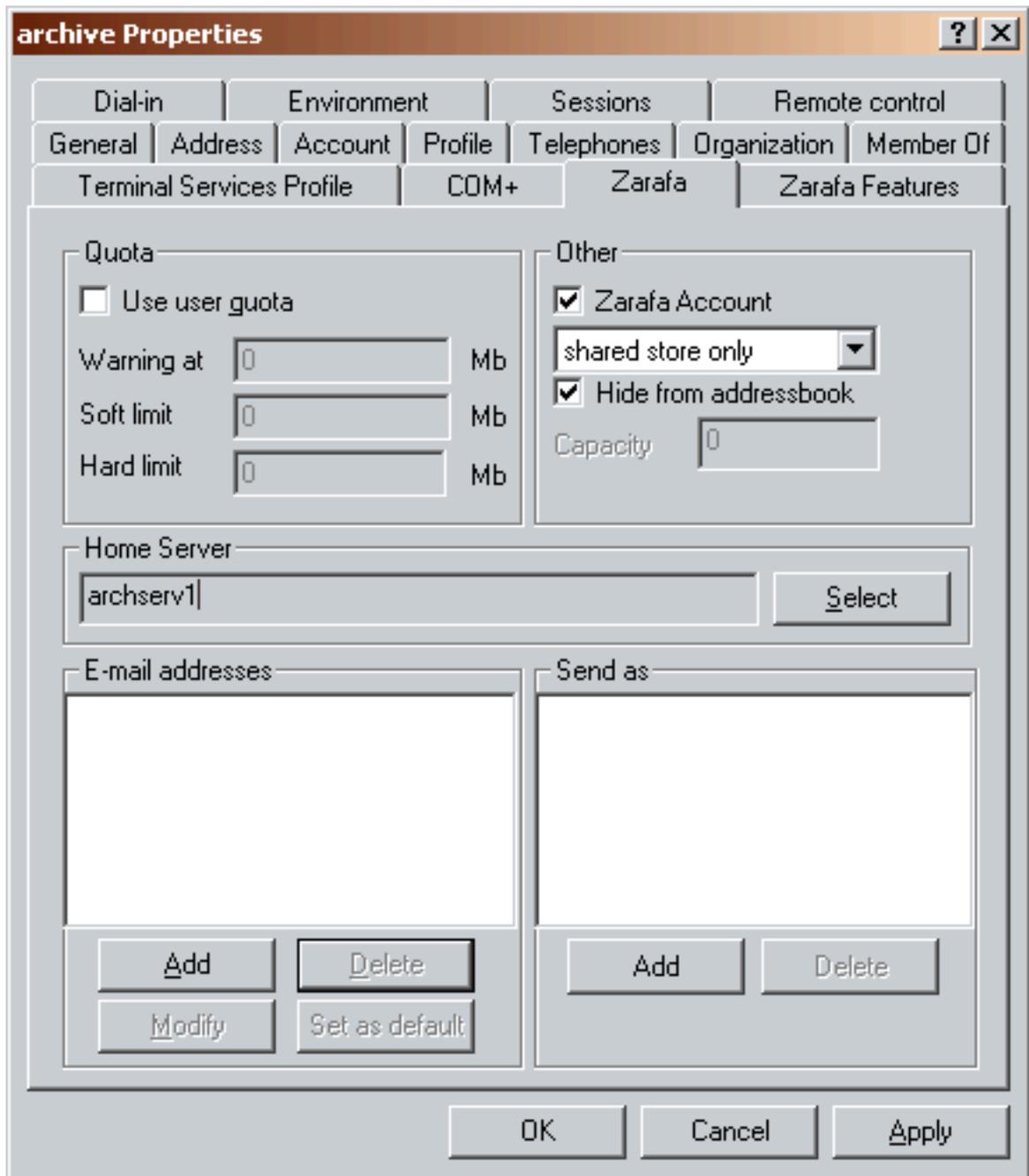
Figure 7.3. Creating an archive store in ADS

7.2.2. Attaching an archive

After an archive mailbox is created, a user can be attached to the archive mailbox. The simplest way to attach an archive mailbox to a user is as follows:

```
zarafa-archiver -u <user name> --attach-to <archive store>
```

This causes a folder with the full name of **user name** to be created in the archive store. This folder is attached to the primary store for **user name** and used as the root of the archive, see screen shot.
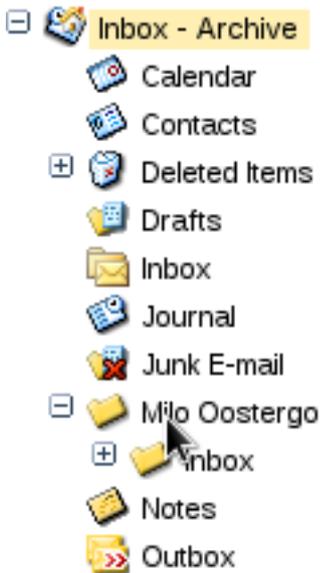
Figure 7.4. Archive store folderlist

A user will have by default read-only permissions on the archive store. To change the default permissions on the archive store, use the **-w** option when attaching the store.

```
zarafa-archiver -u <user name> --attach-to <archive store> -w
```

> **Note**
>
> When multiple users are connected to a single archive store. A user will **only** have access to his/her personal archive and not to the other archives.

If desired, the automatic folder name can be overridden by providing an alternate name:

```
zarafa-archiver -u <user name> --attach-to <archive store> \
        --archive-name <folder name>
```

This causes a folder with the name **folder name** to be created in the archive store. This folder is attached to the primary store for **user name** and used as the root of the archive.

In a one-to-one archive configuration, it's usually not desired to create a folder in the archive store at all. In this case the creation of a folder can be inhibited:

```
zarafa-archiver -u <user name> --attach-to <archive store> --no-folder
```

This causes the root of the archive store to be attached to the primary store for **user name**.

In a single-server configuration, **zarafa-archiver** will not be able to connect to the correct archive server based on the archive store name. In this case the full path to the archive server must be specified:

```
zarafa-archiver -u <user name> --attach-to <archive store> \
        --archive-server <full server path>
```

The full server path is in the form **http[s]://<address>:<port>/zarafa**.

7.2.3. Listing attached archives

To see which archives are attached to a users primary store execute the following command:

```
zarafa-archiver -u <user name> --list
```

This will output a list of attached archives. Each line contains the name of the archive store, the name of the folder that acts as root of the archive and the access rights the owner has in the archive.

The store name will equal the name that was passed when the archive was attached. The archive folder name will be one of three things. It will be the full name of the user if no name was specified when the archive was attached. If a name was specified, the archive folder will be that name. Or if the archive was attached in a one-to-one configuration, it will be *Root Folder*.

The access rights will be *Read Only* when the archive was attached without write permissions. If it was attached with write access it will be *Read Write*. If the permissions were manually changes since the archive was attached, the rights field will display the appropriate role for the rights or all the rights if no matching role exists.

7.2.4. Detaching an archive

Normally an archive can be detached with the following command:

```
zarafa-archiver -u <user name> --detach-from <archive store>
```

If an entry for the archive store is found in the list of attached archives in the primary store, that entry will be removed.

In the rare occasion where a user has multiple attached archives from the same archive store, **zarafa-archiver** will not be able to determine which one the detach from. In that case the folder name also needs to be specified:

```
zarafa-archiver -u <user name> --detach-from <archive store> \
        --archive-name <folder name>
```

Alternatively an archive can be detached by specifying the archive number. This number can be obtained by first listing the attached archives for the user. This method can also be used to detach automatically attached archives. However, **zarafa-archiver** will reattach it on the next run.

```
> zarafa-archiver -u user -l
User 'user1' has 2 attached archives:
    0: Store: Inbox - archive, Folder: user1-archive, Rights: [Read Only]
    1: Store: Inbox - archive new, Folder: user1-archive, Rights: [Read Write]

> zarafa-archiver -u user -D 0
Successfully detached archive.

> zarafa-archiver -u user -l
User 'user1' has 2 attached archives:
    0: Store: Inbox - archive new, Folder: user1-archive, Rights: [Read Write]
```

> **Note**
>
> When detaching an archive that already contained archived and stubbed messages, the stubbed messages can still be opened.

7.2.5. Listing users that have an archive attached

To see which users have an archive attached execute the following command:

```
zarafa-archiver --list-archiveusers
```

7.2.6. See details of archive store

The details of an archive store can be obtained in two ways:

• By requesting the details of the user owning the archive:

```
> zarafa-admin --details user1
Username:          user1
Fullname:          User 1
Emailaddress:      user1@cluster.sio2
Active:            yes
Administrator:     no
Address book:      Visible
Auto-accept meeting req:no
Home server:       cnode-1
Last logon:        12/09/2011 03:41:32 PM
Last logoff:       12/09/2011 03:41:32 PM
Mapped properties:
    PR_GIVEN_NAME           User
    PR_SURNAME              One
    PR_EC_ENABLED_FEATURES  pop3
    PR_EC_DISABLED_FEATURES imap
    PR_EC_ARCHIVE_SERVERS   cnode-2
Attached archives:  1
    Root Folder in Archive - User 1 [Read Only]
Quota overrides:    no
Warning level:      unlimited
Soft level:         unlimited
Hard level:         unlimited
Current store size: 14.86 MiB
Groups (1):
    Everyone

Archive details on node 'cnode-2':
Current store size: 114.68 MiB
```

All attached archive details are appended at the end.

• By explicitly requesting the archive details on a specific node:

```
> zarafa-admin --details user1 --type archive --node cnode-2
Current store size: 114.68 MiB
```

> ★ **Important**
>
> The node on which the archive is stored must be passed in order to find the archive.

These methods only apply to genuine archive stores. Details of regular stores that are manually attached as archives can be obtained by obtaining the details of the user owning that store:

```
> zarafa-admin --details archive
Username:          archive
```

```
Fullname:          Archive Store
Emailaddress:      archive@cluster.sio2
Active:            no
Administrator:     no
Address book:      Hidden
Auto-accept meeting req:no
Home server:       cnode-2
Last logon:        12/09/2011 03:41:32 PM
Last logoff:       12/09/2011 03:41:32 PM
Mapped properties:
    PR_GIVEN_NAME         Archive
    PR_SURNAME            Archive
    PR_EC_ENABLED_FEATURES  pop3
    PR_EC_DISABLED_FEATURES imap
Quota overrides:   no
Warning level:     unlimited
Soft level:        unlimited
Hard level:        unlimited
Current store size: 114.68 MiB
Groups (1):
    Everyone
```

7.2.7. Unhooking an archive store

An archive store can be unhooked the same way as a regular store, but with the addition of the **type** and **node** arguments:

```
> zarafa-admin --unhook-store user1 --type archive --node cnode-2
Store unhooked.
```

7.2.8. Hooking an archive store

An archive store can be hooked the same way as a regular store, but with the addition of the **type** and **node** arguments:

```
> zarafa-admin --list-orphans --node cnode-2
Stores without users:
    Store guid                        Guessed username    Last login      Store size  Store
 type

 -------------------------------------------------------------------------------------------
    F1A6BFCD67604B0FB733F746F1D00A91    user1               <unknown>       0
 archive
> zarafa-admin --hook-store F1A6BFCD67604B0FB733F746F1D00A91 -u user1 --type archive --node
 cnode-2
Store hooked.
```

7.2.9. Removing an archive store

An archive store can be removed the same way as a regular store, but with the addition of the **type** and **node** arguments:

```
> zarafa-admin --unhook-store user1 --type archive --node cnode-2
Store unhooked.
> zarafa-admin --list-orphans --node cnode-2
Stores without users:
    Store guid                        Guessed username    Last login      Store size  Store
 type

 -------------------------------------------------------------------------------------------
```

```
     F1A6BFCD67604B0FB733F746F1D00A91     user1                 <unknown>        0
 archive
> zarafa-admin --remove-store F1A6BFCD67604B0FB733F746F1D00A91 --node cnode-2
Store removed.
```

Running archiver

8.1. Automatic attaching and detaching

Automatic attaching and detaching can be done for all users or on a per-user basis.

8.1.1. From the command-line

The following command automatically attaches and/or detached archives to or from all users.

```
zarafa-archiver --auto-attach
```

It's also possible to explicitly specify for which user to automatically attach and/or detach archives to or from.

```
zarafa-archiver -u <user name> --auto-attach
```

8.2. Perform archiving task

Archiving can be done for all users, all users on one primary server or on a per-user basis.

8.2.1. From the command-line

The following command performs one archive run for all users:

```
zarafa-archiver -A
```

Passing the **--local-only** option to **zarafa-archiver** causes it to only archive the primary stores that live on the server to which **zarafa-archiver** is connected. This is the server on which **zarafa-archiver** is executed unless otherwise configured in the configuration file.

```
zarafa-archiver -A --local-only
```

It's also possible to explicitly specify which users primary store to archive:

```
zarafa-archiver -u <user name> -A
```

It's recommended to perform the archiver run every night. This can be done by adding the following line to **/etc/crontab**.

```
0 1 * * * root [ -x /usr/bin/zarafa-archiver ] && /usr/bin/zarafa-archiver -A
```

8.3. Perform cleanup task

To keep the archive in sync with the primary store when items are deleted in the primary store, the archive needs to be started in cleanup mode. This is a separate operation because it's a time consuming operation. Therefore it's advisable to run cleanup less often than the archive operation.

8.3.1. From the command-line

The following command performs a cleanup for all users:

```
zarafa-archiver -C
```

Passing the **--local-only** option to **zarafa-archiver** causes it to only cleanup the archives of users who have a store on the server to which **zarafa-archiver** is connected. This is the server on which **zarafa-archiver** is executed unless otherwise configured in the configuration file.

```
zarafa-archiver -C --local-only
```

It's also possible to explicitly specify which users archive to cleanup:

```
zarafa-archiver -u <user name> -C
```

It's recommended to perform the cleanup run once a week. This can be done by adding the following line to **/etc/crontab**.

```
0 3 * * 0 root [ -x /usr/bin/zarafa-archiver ] && /usr/bin/zarafa-archiver -C
```

Client Features

To allow a user to browse and search in his archive or in the archives of shared stores, the archives are automatically added in Outlook and the Webaccess.

> **Note**
>
> This only applies to auto attached archive stores. Regular archives can be opened as a shared store manually.
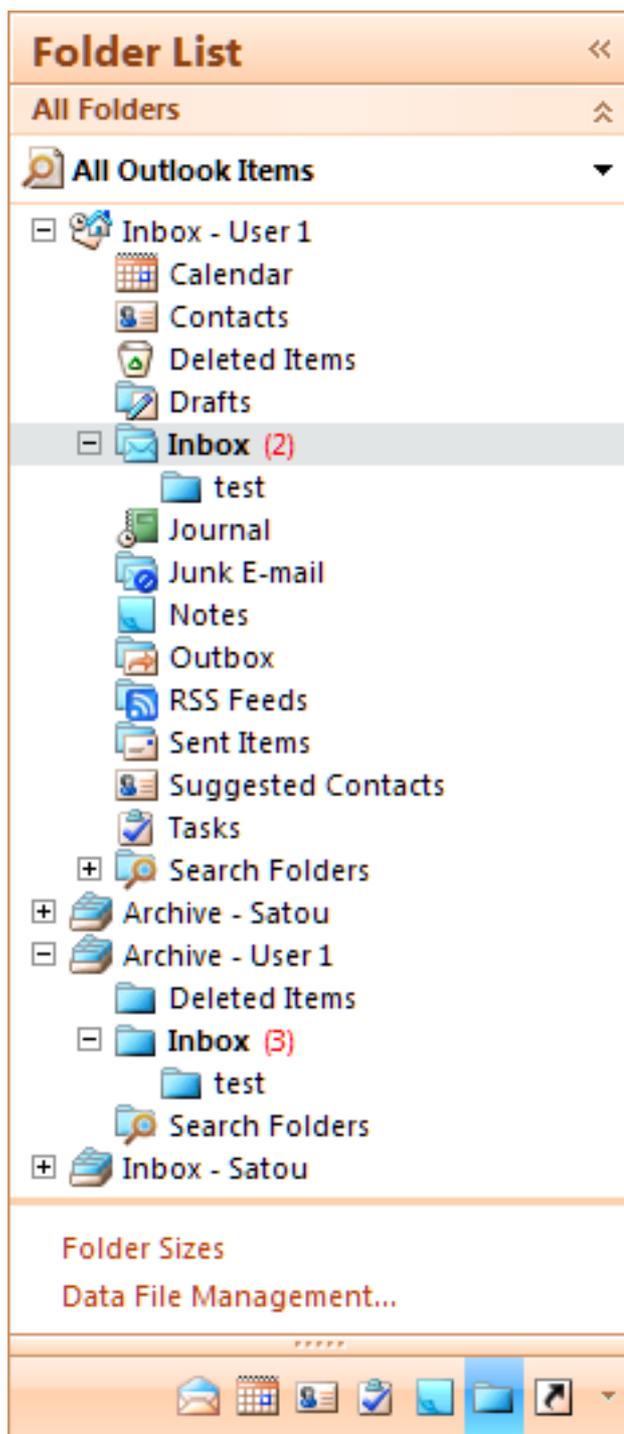
Figure 9.1. Outlook Hierarchy with Archives
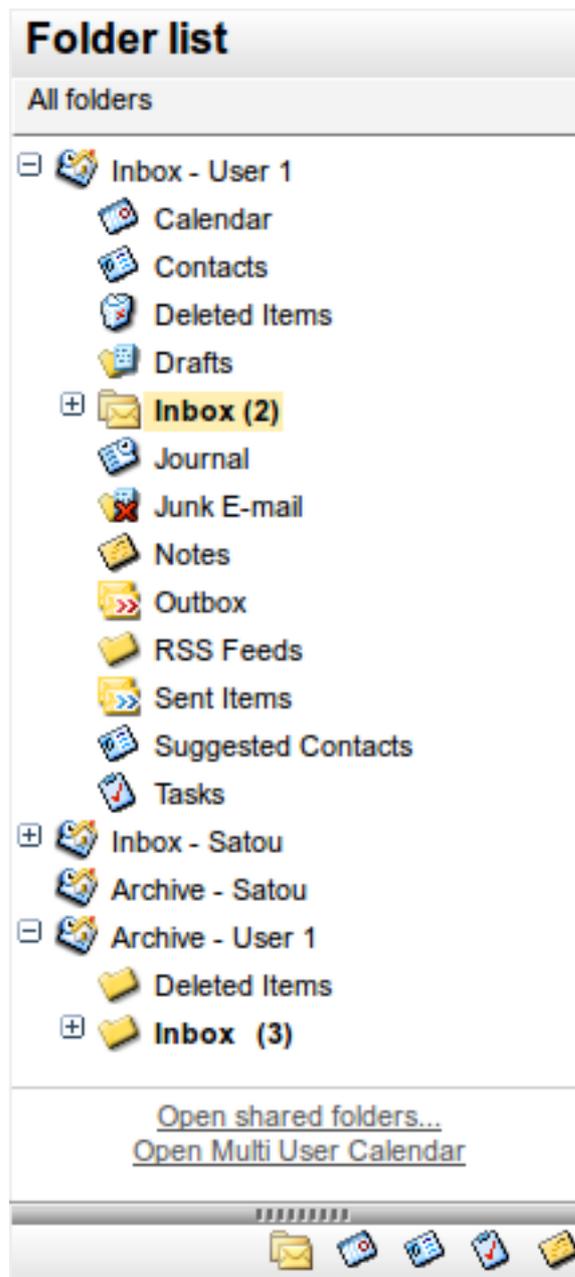
Figure 9.2. Webaccess Hierarchy with Archives

Multi Single Server Setup

In the multi single server setup, the zarafa-archiver in configured in such a way that the archive resides on another Zarafa server, which is *not* part of the same cluster as the primary server. Usually this means that none of the Zarafa servers is configured for multi-server usage.

The advantage of this setup is that no multi server license is required. This comes at a cost: stubs are not supported and the archive store can't be added as a shared store in an existing profile.

Because the archive server is not known by the primary server, the archive server must be specified when the archive is attached to a user store.

```
zarafa-archiver -u user --attach-to archive --archive-server https://archive.zarafa.com:237/
zarafa
```

This attaches the user **user** to the store called **archive** on the server that can be connected to through **https://archive.zarafa.com:237/zarafa**.

Important

The archive server must be accessed through SSL in order to gain the correct access level.

Note

The **--archive-server** option is only useful in this setup.

Zarafa Archiver Extras

The Zarafa Archiver Extras package contains additional tools to enhance the basic Zarafa Archiver functionality.

11.1. Installation

The Zarafa Archiver Extras can be found in the zarafa-archiver-extra package.

11.2. The Tools

11.2.1. Zarafa Archiver ACL Sync

11.2.1.1. Description

`za-aclsync` Synchronizes archive ACL settings with those of the primary store.

When a user has set permissions for other users or groups on his or her store or folders, the other users will need at least read-permissions on this persons archive as well in order to read stubbed messages or access the archive directly. These permissions can not be set by the owner of the archive if the archive was attached without write privileges. Even if the user has write permissions, it's a nuisance to have to set all the permissions twice (or more if multiple archives are attached).

`za-aclsync` can be used to propagate the ACL settings from the primary store to the archives. However, no user will ever get more right on a store or folder than the owner of the archive. So if the archive was attached without write permissions, no user will get write permissions on the archive.

For every archived folder in an archived store `za-aclsync` will first determine the rights for the owner of the archive. Then it will get all the entries from the ACL of the current folder except those of the owner. Each right will be masked with the rights of the owner before being added to the ACL of the archive folder.

11.2.1.2. Usage

```
za-aclsync [options] [users]

options:
-h serverpath           : Host to connect with.
-s sslkey_file          : SSL key file for authentication.
-p sslkey_pass          : Password for the SSL key file.
```

`users` is a space separated list of users for which to synchronize the ACL settings. If no user is specified all users will be processed.

11.2.2. Zarafa Archiver ACL Set

11.2.2.1. Description

`za-aclset` sets or updates the permissions on an archive store for the owner of that store.

When an archive is attached to a store, the owner of the store get read-only or read/write permissions on the archive based on the configuration or command line options passed at the time of attaching.

Also when an archive is attached to a store of a non-active user, no permissions for that non-active user are set because that's not possible.

In the first case it can be desirable to change the permissions at a later time because of company policy changes or because the original setting was just wrong.

In the second case, if the user is converted to an active user, it's required to reset the permissions because the owner won't have any permissions on his archive.

11.2.2.2. Usage

```
za-aclset [OPTIONS] [users...]

options:
-h | --host         Host to connect with. Default: file:///var/run/zarafa
                    Three formats are allowed for this option:

                      UNIX socket : file://<path to the UNIX socket>
                             HTTP : http://<host or IP>:<port>/zarafa
                      Secure HTTP : https://<host or IP>:<port>/zarafa

-s | --sslkey-file  SSL key file for authentication.
-p | --sslkey-pass  Password for the SSL key file.
-w |                Grant write permissions on the archive.
--writable <y|n>    Enable or disable write permissions.
--help              Show this help message.
```

users is a space separated list of users for which to synchronize the ACL settings. If no user is specified all users will be processed.

11.2.3. Zarafa Archiver Restore

11.2.3.1. Description

za-restore can be used to repopulate a primary store to a state where no archive is required to read any message. In a less cryptic way this means that all stubbed messages are destubbed and all messages that were deleted after archiving are restored.

The advantage of using **za-restore** over dragging the messages back from the archive in Outlook or Webaccess is that the restores messages are sanitized, allowing them to be properly re-archived later.

11.2.3.2. Usage

```
Usage: za-restore [OPTIONS] user
OPTIONS:
  -h | --host           : Host to connect with. Default: file:///var/run/zarafa
  -s | --sslkey_file    : SSL key file for authentication.
  -p | --sslkey_pass    : Password for the SSL key file.
  -l | --log-file       : Select log file.
  --detach              : Detach the selected or all archive stores before
                            starting the restore procedure. This avoids the
                            archiver from re archiving restored messages.
  --unhook              : Unhook the selected or all archive stores once
                            the restore process has completed. This implies
                            --detach and only works on archive stores.
  --remove              : Remove the selected or all archive stores once
```

```
                             the restore process has completed. This implies
                             --unhook and only works on archive stores.
  --select-source      : Select the source archive(s) by providing a comma
                             separated list of archive indexes. The indexes specify
                             which archives to restore from. The --detach, --unhook
                             and --remove options only apply to the selected
                             archives.
                             The archive indexes can be obtained by listing the
                             attached archives for a user: zarafa-archiver -u
                             <user> -l.
  -v | --verbose       : Increase console loglevel. Can be specified multiple
                             times.
  -q | --quiet         : Decrease console loglevel. Can be specified multiple
                             times.
  -N | --dry-run       : Don't actually modify anything.
  --help               : Show this help message.
```

11.2.3.3. Example

The following example will completely restore the store of john_doe and detaches and unhooks all archive stores while logging to **/tmp/john_doe_restore.log**

```
> za-restore --unhook -s /etc/zarafa/ssl/archiver.pem -p password \
  -l /tmp/john_doe_restore.log john_doe
```

Note that no host is specified, causing za-restore to connect to **file:///var/run/zarafa**. The sslkey_file and sslkey_pass are specified in order to connect to the other nodes in the cluster.