AIS Storage user guide

Xavier Tordoir

December 22, 2011

Contents

Part I

Prerequisites

Chapter 1

Definitions and concepts

The AIS storage system is a cloud based storage system. As such, storage operations are performed by HTTP requests exclusively, either through the web application or the REST API, the latter operation being wrapped in java and python libraries. The system is very similar to the Amazon S3 or Google Storage products. The main features differentiating these products from our solution is that the physical storage is provided by third parties and not exclusively by us.

1.1 bucket

A Bucket is a container for files and objects. The user can consider a bucket as a kind of virtual disk or workspace. Buckets cannot be nested, they are always the root of a set of data to be organized within the bucket. Sharing data with collaborators and customers is configured at the bucket level, all objects within a bucket share the same access rules. The user can so consider a bucket as a single project allowing to grant access on its content to other selected users.

1.2 bucket metadata

Bucket Metadata are sets of key-value pairs associated with a bucket. These metadata allow the owner of a bucket to configure the behaviour of the bucket with respect to data storage (see ...), and to add some general information about the bucket.

1.3 object

Objects are the elements contained in a bucket, no object can exist outside of a bucket. All objects have a *key* with is a text uniquely identifying the object within the bucket. In a simple usage scenario, the *key* is the path of a file or directory within the bucket.

An object is actually a pointer to a unit of data. This object can point to a file, a directory or any other type of data that can be stored in the form of object metadata.

1.4 object metadata

Metadata in the form of key-value pairs can be associated with objects. This mechanism is used for example to identify an object as a file on a physical storage device. These metadata allow to configure the behaviour of an object (for example a file can be downloaded), or for the user to add some general information about the object.

1.5 store

While the list of objects stored in buckets as well as metadata are maintained in a database, files are stored on physical disks. The system is able to access these physical files through the concept of stores. A store is an ssh connection to a server providing storage space. This connection is a username, a server address (IP or dns) and a root directory under which all data are stored.

Part II

User guide: web interface

Chapter 2

Store management

The use of the storage system relies on a database used to store the buckets and objects definitions and metadata on one side and on a number of *store* to store the files on the other side. The database is provided by the AIS storage middleware, while the stores are dynamically plugged on the system and access is granted to users. So store management is the basics to be able to manage files in the system.
Most users will not actively manage stores, but will simply use them, as storage backend for their files, in a transparent way. So this chapter is intended for physical storage providers or users wanting to know more about the file storage details.

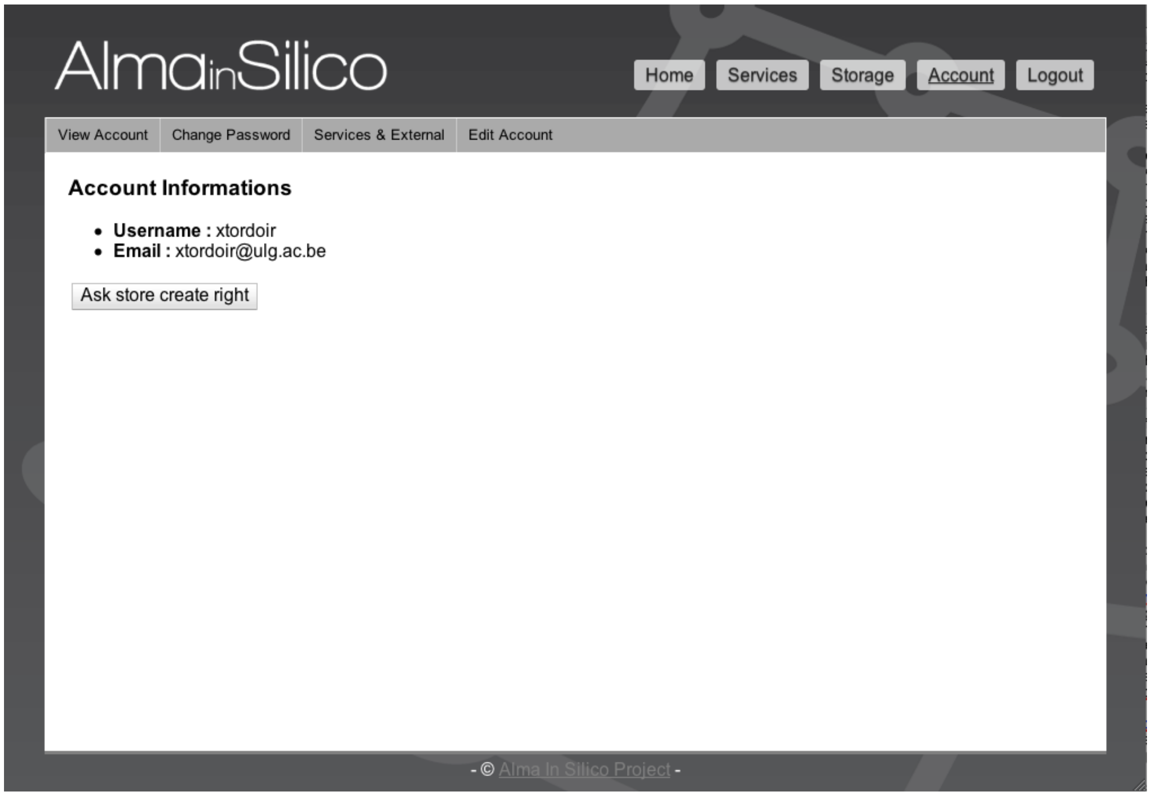
2.1 Store creation

The creation of a *store* is reserved to users with the role allowing them to manage stores. This role can be requested on the account tab on the web application. As mentioned in the definition of store, a store is an ssh connection to a file server. With this right, a stores can be created.

2.1.1 Store definition

The minimal information required to create a store consists in the following fields:

- namespace: a unique string identifier for the store, 8 characters maximum, alphanumeric
- username: the username to be used to login on the remote server
- store_url: the server IP or dns
- path: the absolute path on the server of the data directory



Account Informations

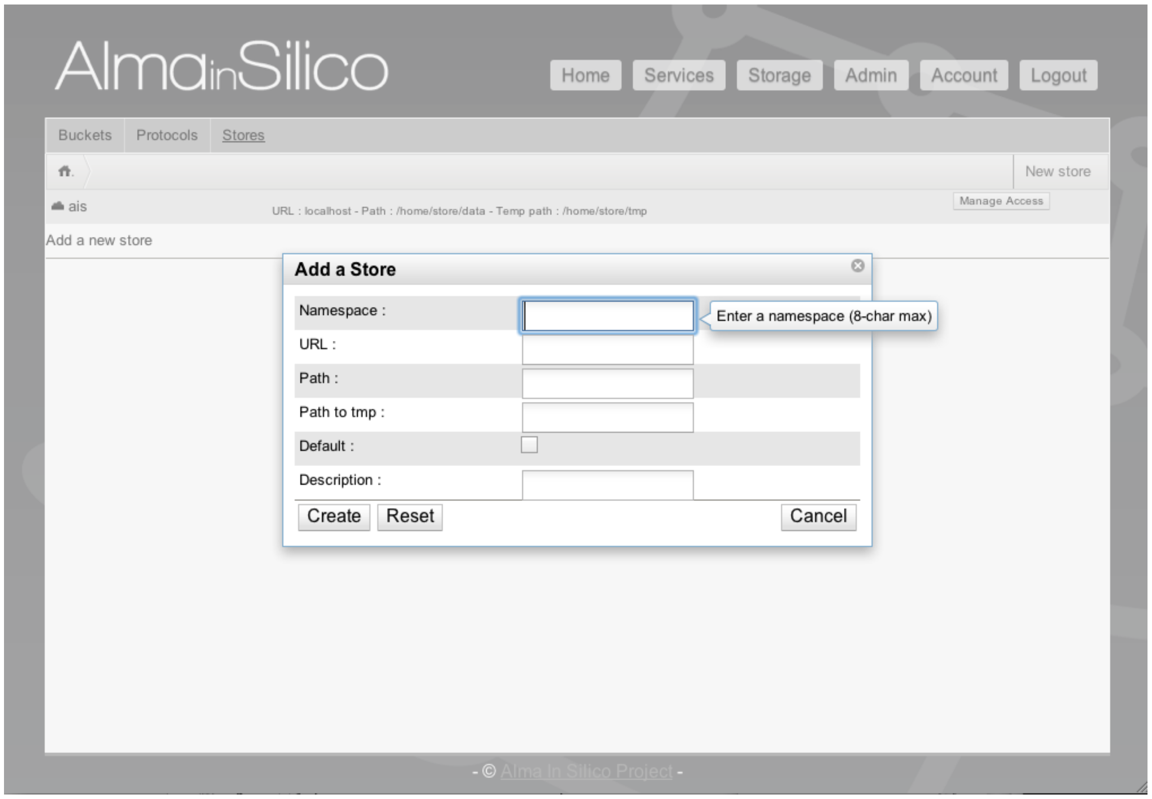
- **Username :** xtordoir
- **Email :** xtordoir@ulg.ac.be

Figure 2.1: Request store creation rights

- tmp: the absolute path on the server of the temporary data, it is used during local copy processes (see ...)

- description: the text description of the store

Store creation is done exclusively on the web application, in the storage tab under the store panel: Once a store is created it is not readily available. Of course, the AISStorage web



ais URL : localhost - Path : /home/store/data - Temp path : /home/store/tmp

Add a new store

Add a Store

Namespace :	Enter a namespace (8-char max)
URL :	
Path :	
Path to tmp :	
Default :	☐
Description :	

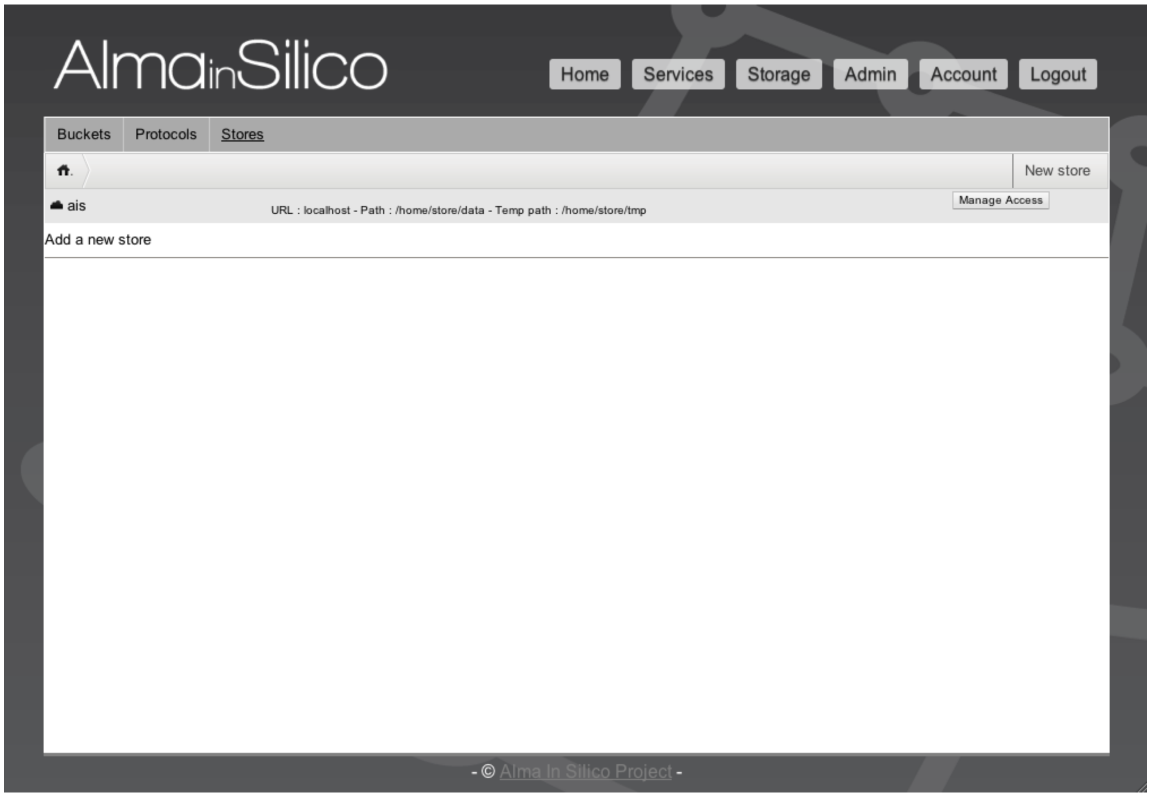
Figure 2.2: Store creation form

server must be able to access the remote file server. The authentication consists in sharing the ssh public key of the AIS server with the file server.

2.1.2 Store validation: AIS public key installation

For the AIS storage system to work with a store, it must be able to connect to the store. To allow this connection, the store manager must download the AIS rsa public key and install it in its authorized_keys file. The public key is downloaded from the stores page. The content of this file must be concatenated to the content of the *~/.ssh/authorized_keys*

file on the store.
After completion of this operation, the *Put online* button on the store list can be pushed to validate the connection. By this action, the store manager actually asks the AIS server to try to connect to the store and check that the store is correctly configured: it checks if the data and tmp directories exist and have the correct access rights (must be writable) or tries to create them. If these operations are a success, the store is online and can be accessed by users, or put offline. If the store cannot be put online, the manager is notified and can try again after adjusting the store or server configuration.



URL : localhost - Path : /home/store/data - Temp path : /home/store/tmp

Figure 2.3: Download the public key, then put the store online

2.2 Store rights management

The creator of a store is granted administration rights and is able to grant rights on the store to other users. Some right can be applied to all users at once, these public rights are:

- visibility: control whether the store is visible to all users. If yes, all users in the system are aware that the store exists, although they don't have access to the store, they are able to make a request to access it, directly from the store list. The store administrator will accept or reject such requests.

- writable for all: this controls whether all users have a write access on the store. This only works if the store is visible.

The rights granted to users (by email address) are:

- Admin: read and write access, administration rights (users management)
- Users: read and write access, can download and upload files on the store
- viewer: read access, can download files from the store

To grant a right to one or several users, email addresses must be entered in the field and validated.

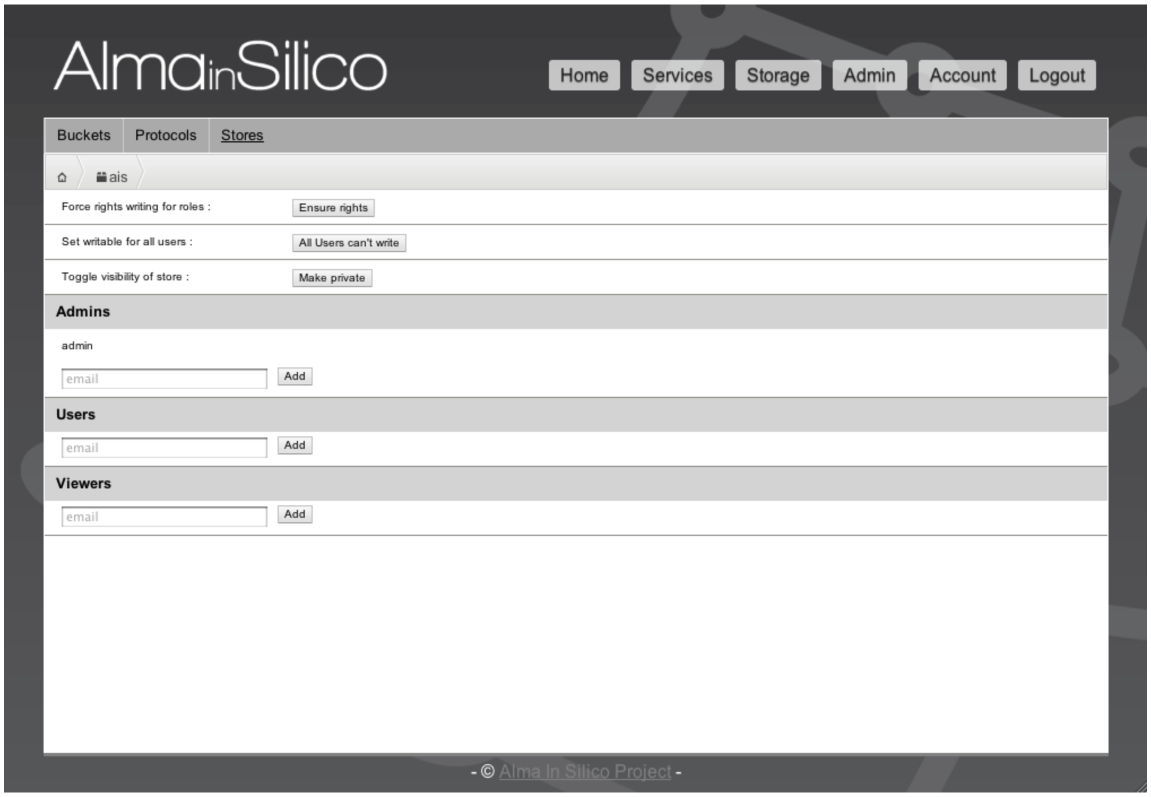


Figure 2.4: Store rights management page

Chapter 3

Bucket management

All users have the ability to create and manage their own buckets.

3.1 Bucket creation

3.1.1 Bucket definition

In order to create a bucket, the minimal information required are the following mandatory fields:

- name: the name of the bucket, unique across all users buckets.
- description: a text description of the bucket

Other available options for advanced configuration can be selected. These control the details on how files are organized on the different stores. As these options cannot be changed, it is important to understand what they mean if default values are not used. It is advisable to have a reason to select non default values:

- store.multiplicity: the multiplicity of stores used by the bucket, can be *unique* or *multiple*. A multiple store allows to store files on multiple devices, it is the default value. Multiple devices allows to spread the data over different servers, it is meaningful different data are generated "close" to different stores and file transfer shouldn't be performed unnecessarily. A unique store is selected when a tight control on files location must be enforced, and all data should be physically located in a single place, for example close to a data processing facility.
- store.default: the default store, this option is mandatory if store.multiplicity is *unique*.

- fs.mapping: the filesystem mapping type, can be either *mirror* or *tree*. The mirror mapping means that object keys are paths on the physical storage. Such a path is relative to the store data directory path. The default is *mirror* as it allows to manage data as in a filesystem. This ensures that all files are at leaves in the path tree and that exporting a bucket on a local filesystem is simple. On the other hand, it is not possible for an object to point to a file from an other bucket, also, a file cannot have child nodes.

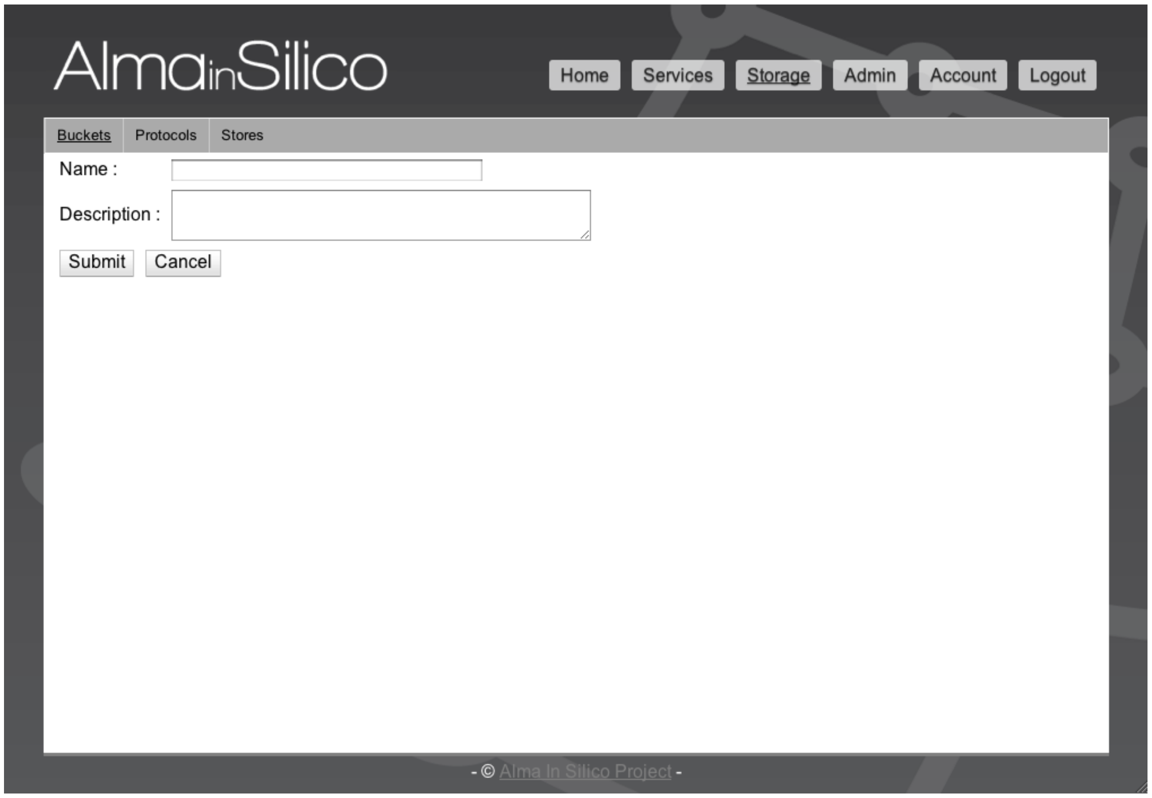


Figure 3.1: Bucket creation form

Once created, the bucket is readily available for use to its creator.

3.1.2 Bucket metadata

While buckets metadata are supported and can be queried and set with the REST API, no view is available from the web UI.

3.2 Bucket rights management

The creator of a bucket is granted administration rights and is able to grant rights on the bucket to other users. A right can be applied to all users at once, this public right is:

- visibility: control whether the bucket is visible to all users. This allows users to publish data to the public. Also, it provides users with the possibility to request write access to the bucket, this is done directly from the list of buckets. Such a request will be accepted or rejected by the bucket administrator.

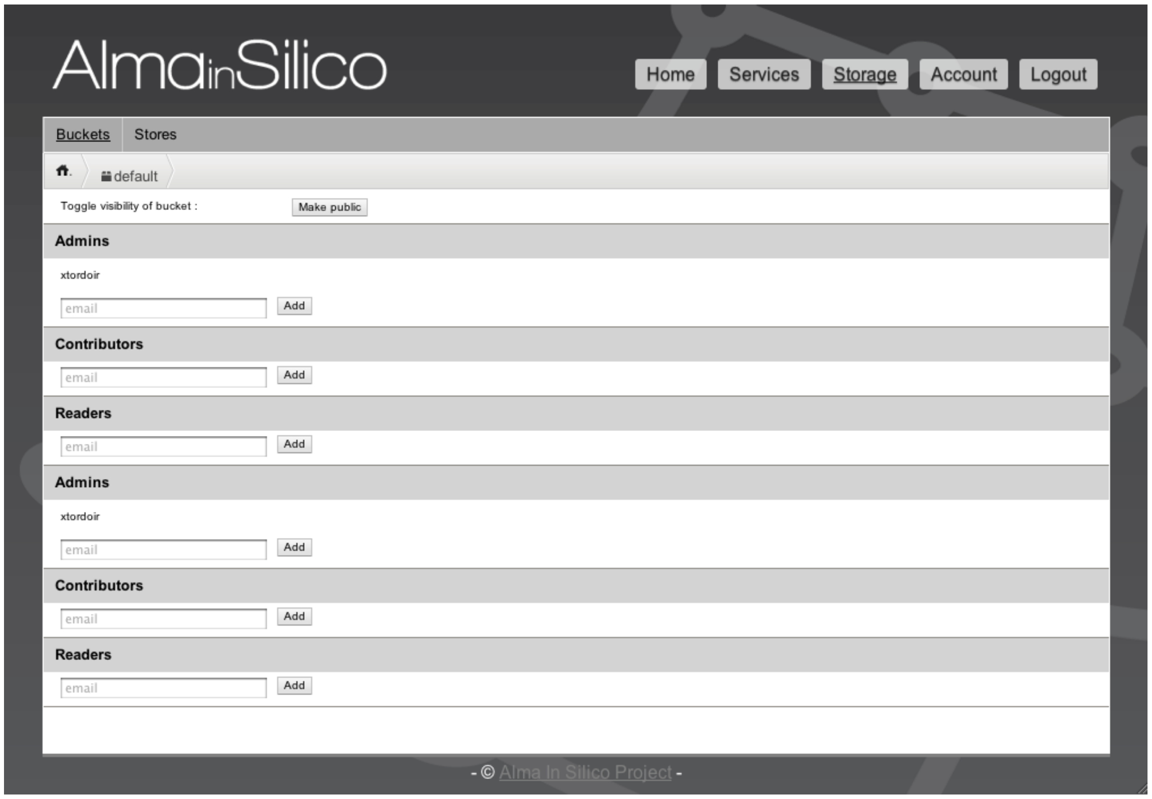
The rights granted to users (by email address) are:

- Admin: read and write access, administration rights (users management)
- Users: read and write access, can download and upload files on the bucket
- viewer: read access, can download files from the bucket

To grant a right to one or several users, email addresses must be entered in the field and validated.

3.3 Relationship between Store and bucket management

While the user would like to control access rights on a bucket by just tuning the bucket rights management page, the combination of store rights and bucket configuration can result in side effect for the bucket rights.
The rule is that always the store rights have precedence over the bucket rights, and store rights only applies to files. for example, if a user has read access on a bucket, he will be able to see the complete list of file even if they are stored on stores he has no read access to. The limitation in this case is that the user is not able to download the file as he has no read access on the store. The same applies for write access. If a user has write access on a bucket configured to work with a unique store for which no write access is granted, the user will not be able to upload files, while he will be able to create objects.



Toggle visibility of bucket :

Admins

xtordoir

Contributors

Readers

Admins

xtordoir

Contributors

Readers

Figure 3.2: Bucket rights management page

Chapter 4

Object management

From a selected bucket and a given object it is possible to create new objects. From the bucket root, objects can be created in a nested way, and so organized as a tree structure. The '/' characters is used in the object *key* as a separator to jump from one level to the next. The *key* can be seen as a path on the virtual filesystem. For mirrored buckets, the *key* effectively represent a path on the store filesystem. For stores with a *tree* fs.mapping property, there is no guaranteed correspondence between the key and a store filesystem path.
As a result, objects can be seen as directories and files.

4.1 Object creation: make directory

The operation of creating a simple object without associated file in a bucket is equivalent to creating a directory. There is no need to create parent directories for this operation to work, and the operation will not create objects for the virtual parent directories. The text representation of the object key is sufficient for the system to understand the implied tree structure.

4.1.1 Object definition

The creation of a simple *object* is reserved to users with write access on the target *bucket*. The fields difining a simple object is:

- path: the virtual path or *key* of the object. It is a unique identifier also locating the object on the virtual tree structure, using '/' as directory or node separator.

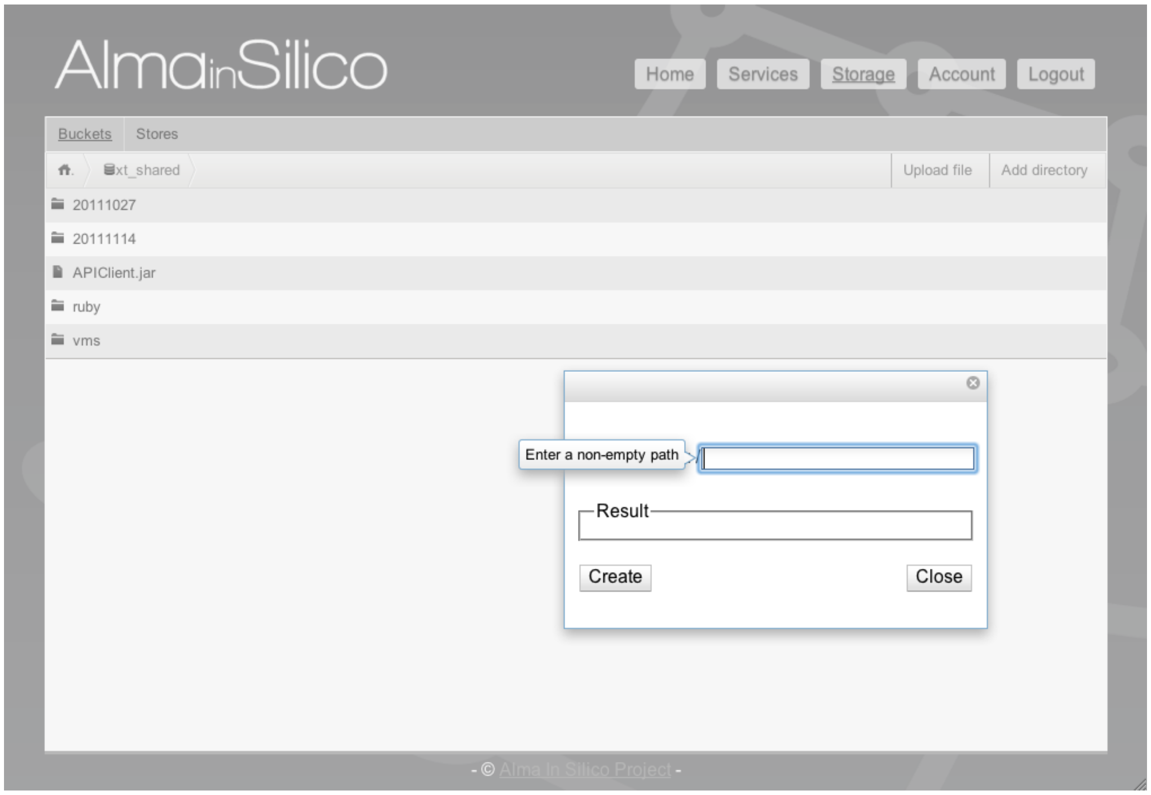


Figure 4.1: Object (directory) creation form

4.1.2 Object metadata

The web interface does not allow for the creation or edition of object metadata by the user. The REST API provides this functionality. The only object metadata created from the web interface are the once that identify an object as a file. The object metadata are nevertheless visible.

4.1.3 Object creation: upload file

The operation of putting a file in a bucket consists in creating an object and associating this object with a file on a store. This operation requires to have write access on both the bucket and the store on which the file will be located.

4.1.4 Object file definition

The definition of a file in the AIS storage is actually an extension of the simple object (or directory) definition. For the creation of a file object, the following fields must be filled-in:

- file: the file path or *key* is constructed from the currently selected object (directory or node) and the filename of the upload item.
- *store*: the uploaded file will be stored on a physical device (a store). Only a store for which the user has write permission can be selected.

4.1.5 Object file metadata

As mentioned earlier, a file object is an extension of a simple object. The extension is made through the use of object metadata, specific for the file object definition. The set of mandatory metadata required to make an object a file object are the following:

- url: a string pointing to the physical location of the file. It consists in the *store* and the path on the store relative to the store's data directory.
- size: the size of the fiel in bytes
- md5: the md5 of the file
- username: the username of the user who created the file in AIS storage. The owner of the file is the person responsible for the storage provider's storage usage.

All these fields are set automatically by the server during the file upload operation.

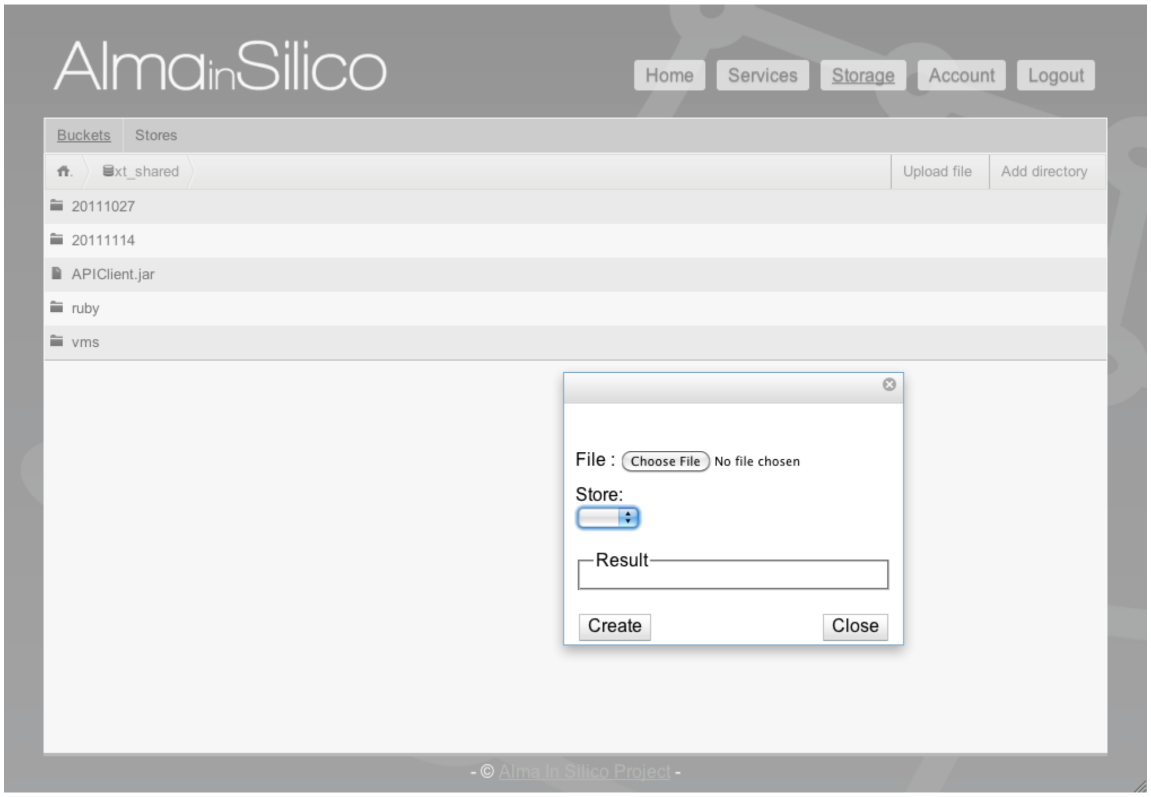
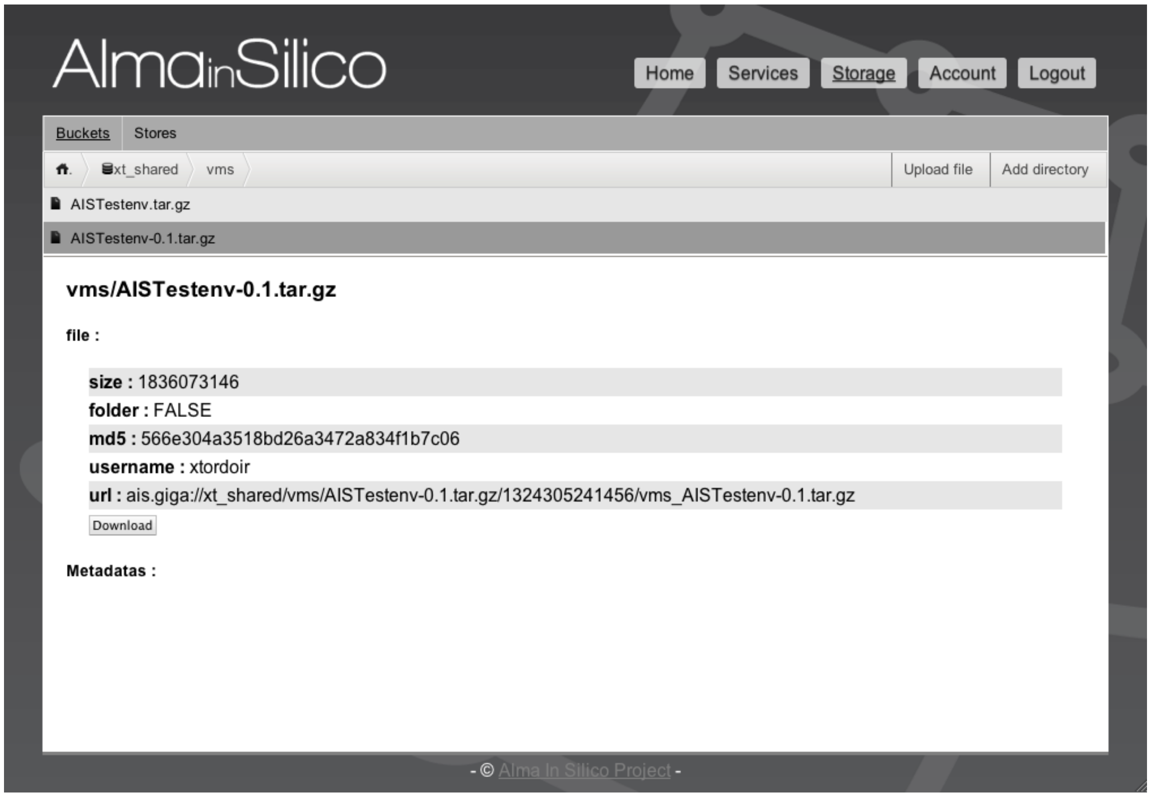


Figure 4.2: Object (file upload) creation form



vms/AISTestenv-0.1.tar.gz

file :

size : 1836073146
folder : FALSE
md5 : 566e304a3518bd26a3472a834f1b7c06
username : xtordoir
url : ais.giga://xt_shared/vms/AISTestenv-0.1.tar.gz/1324305241456/vms_AISTestenv-0.1.tar.gz

Metadatas :

Figure 4.3: File object metadata view

Part III

User guide: REST API

Chapter 5

Store

Stores cannot be managed with the API, only the web interface allows to create and manage access control on stores.

Chapter 6

Bucket

6.1 Bucket creation

6.2 Bucket

6.2.1 Object as File

6.2.2 Object metadata

6.3 Object API