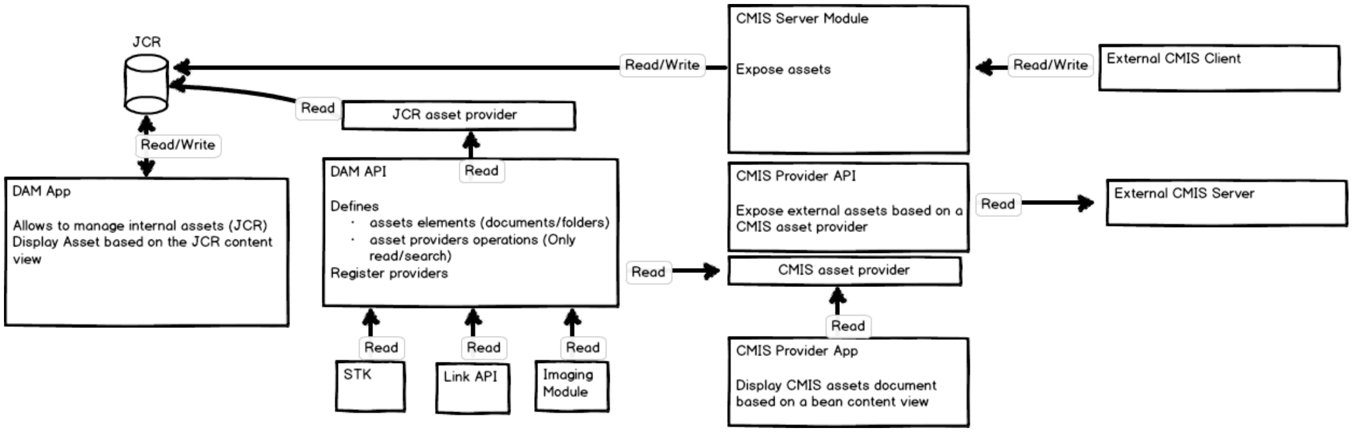
Concept - DAM & CMIS - Project definition

Introduction

The main goal of this project is to expose internal DAM assets to a CMIS client and to refer to assets (documents) exposed by a CMIS server from templates/links,...

The following mockup shows a first high level interactions between the modules:



JCR
Read/Write
Read
JCR asset provider
Read/Write
DAM App
Allows to manage internal assets (JCR)
Display Asset based on the JCR content view
DAM API
Read
Defines
assets elements (documents/folders)
asset providers operations (Only read/search)
Register providers
Read
Read
Read
STK
Link API
Imaging Module
Read
CMIS Server Module
Expose assets
Read/Write
External CMIS Client
CMIS Provider API
Expose external assets based on a CMIS asset provider
Read
External CMIS Server
CMIS asset provider
Read
CMIS Provider App
Display CMIS assets document based on a bean content view

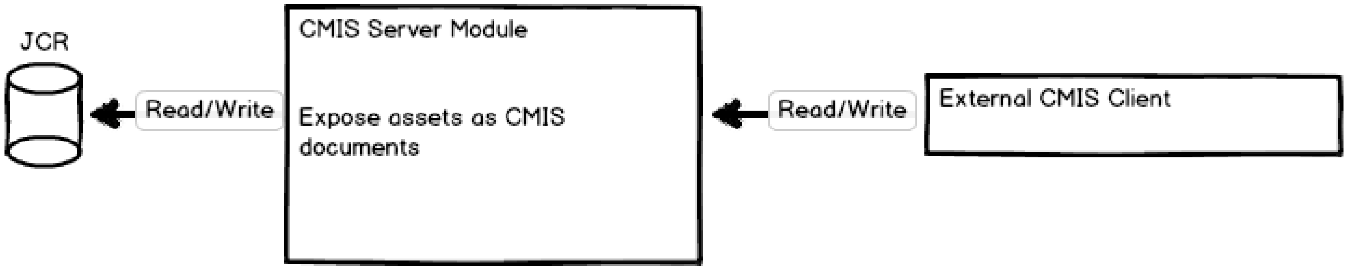
CMIS (*Content Management Interoperability Services*)

CMIS is an open standard that allows different content management systems to inter-operate over the Internet. Specifically, CMIS defines an abstraction layer for controlling diverse document management systems and repositories using web protocols.
In order to expose internal DAM Assets and access external CMIS resources, we have to:

- expose a CMIS server communicating over web service with external CMIS client.
- create a CMIS client module that support both w**eb service and the atom** binding in order to access external CMIS resources.

CMIS Server

This is the M5 implementation of the M4.5 CMIS module.

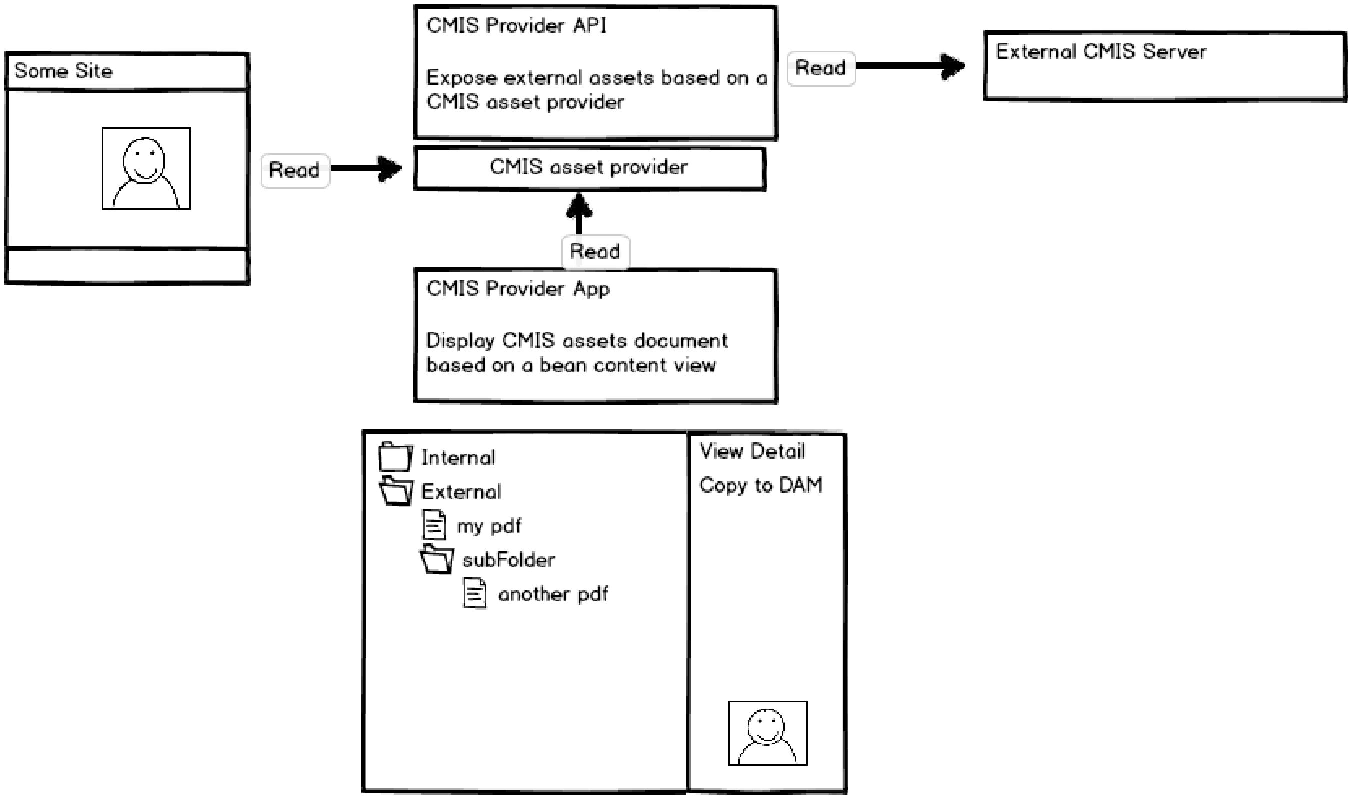


The CMIS server module allows an external CMIS client to brows the internal DAM assert repository, retrieve and create Assets as CMIS documents.

Note that this module directly access the DAM JCR workspace and does (currently) not use the Asset API

CMIS Server detailed concept page

CMIS Provider



The CMIS provider has two parts:

- CMIS Provider API : This sub module implements the DAM API in order to retrieve CMIS documents as assets.
- CMIS Provider App: Define a content App able to display assets in a tree/list/... view.

CMIS Provider API

- Implements the Asset API in order to define a
    - CMIS asset provider
    - Mapping of CMIS documents into assets elements
- Implement/Configure a CMIS client
- Register the CMIS asset provider

CMIS Provider App

- Is JCR Agnostic and based on a bean content app.

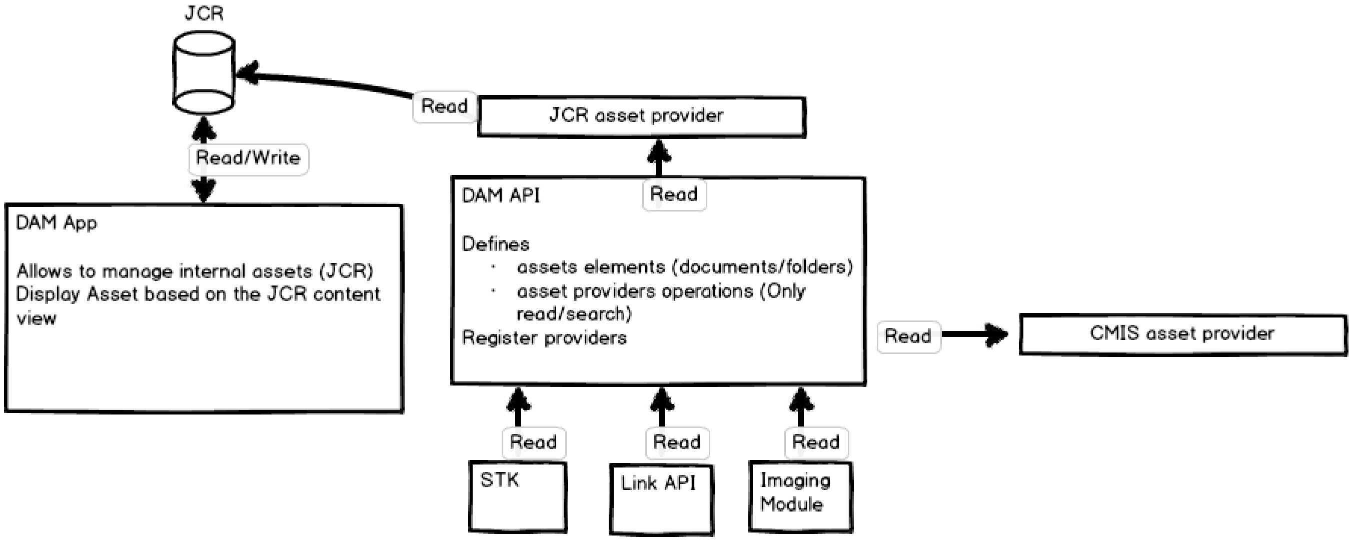
- Displays CMIS documents as assets in tree/list/thumbnail view.
- Defines and configures the content view used in federated choose dialog.
- Allows read operations (Edit assets detail)
- Allows search operations

CMIS Provider module detail concept page

DAM

The two main parts touched by this project are

- DAM App : The app has to define a new choose dialog supporting external providers (like CMIS)
- DAM API: This is the main entry point for the rendering part in order to access internal or external assets in a federated way.


JCR
Read
JCR asset provider
Read/Write
DAM App
Allows to manage internal assets (JCR)
Display Asset based on the JCR content view
DAM API
Read
Defines
· assets elements (documents/folders)
· asset providers operations (Only read/search)
Register providers
Read
CMIS asset provider
Read
Read
Read
STK
Link API
Imaging Module

DAM App

- Define the federated choose dialog
  - Define a tab per registered provider
  - Delegate to the provider app the rendering of the content views (JCR based for Internal assets, CMIS,...)

DAM API

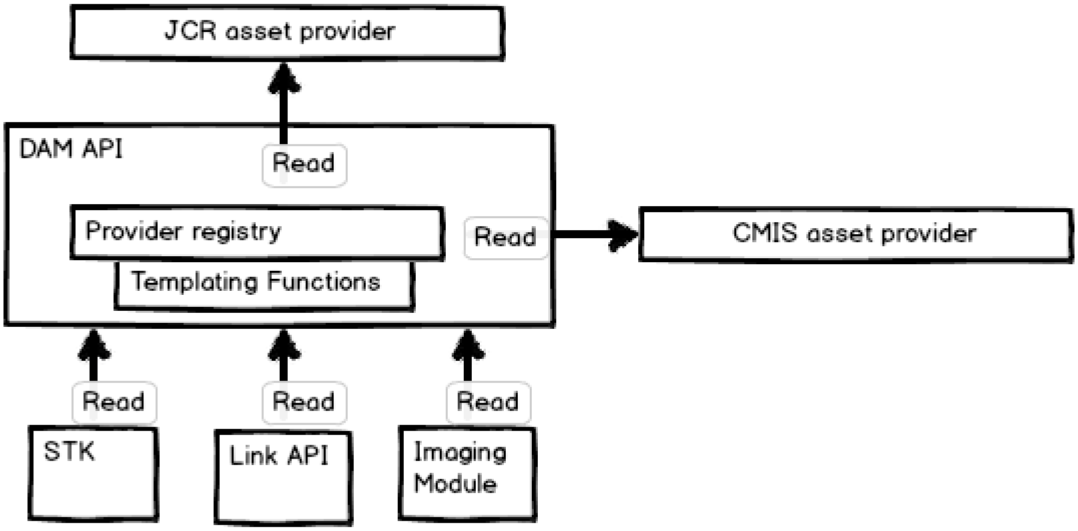
- Define the Asset API
  - Asset elements (folders/documents/Metadata)
  - Asset providers (Access an asset element for read/search operations)
- Register all defined asset providers

External providers have to implement the Asset API in order to expose assets. This API is also used by **content views** in order to display assets in views (list/tree/thumbnail).
Templating/Links uses this asset provider registry to access assets and perform search operations.

DAM Module detail concept page

DAM API Design Notes

Rendering



The rendering part includes STK, the Link API and the Imaging module. These components access the DAM API module in order to resolve/create links, retrieves assets, ...
Imaging module has to support rendition for any asset providers.

Link API detailed concept page

Imaging module detailed concept page

Key features

DAM API

The DAM API will be completely reviewed in order to:

- Support
  - Folders: The DAM API needs to have a concept for folders to make the assets navigable. Folders do not have metadata and cannot be linked to (href links, referencing them is supported) nor do they have renditions.
  - Capability: Not all providers will have the same feature set. We need a mechanism in which users of the API can test for certain capabilities.
  - Querying : We currently have an AssetFilter class with basic properties for searching for specific assets. It contains a string that's used as an addition to the where clause. This will obviously not work for other providers
  - A better rendition process
- Define an Provider's registry acting as entry point for the rendering part
- Have a more lightweight and simpler API

CMIS Provider App

Should base on new jcr agnostic container/presenter etc but there'll most likely be some asset specific things.

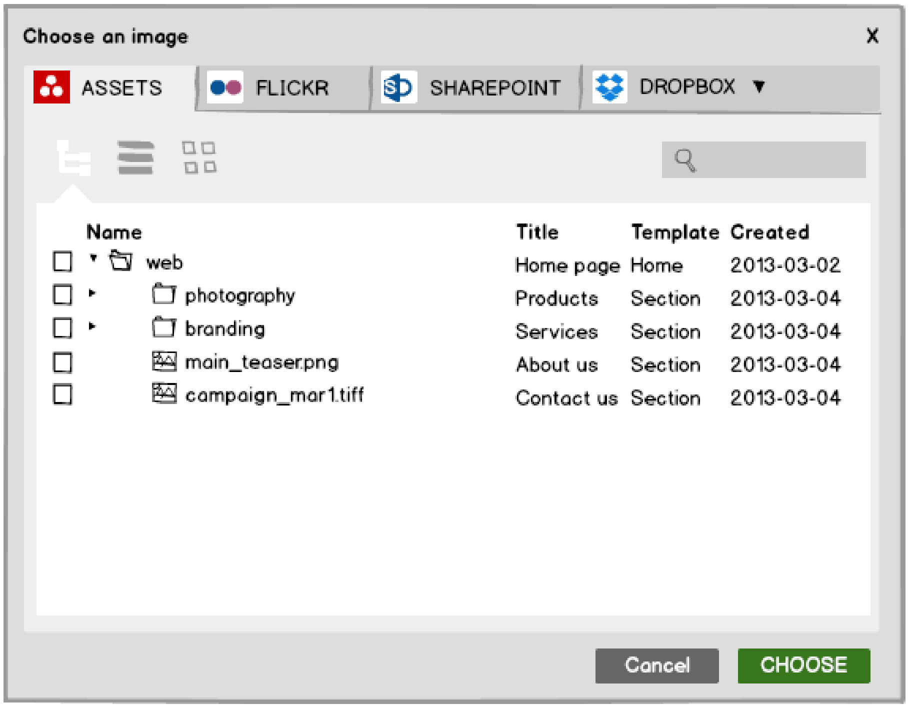
By using at the end a 'AssetProviderContainer' we will have the capability by simply implementing the DAM API (Provider, Asset, ...) for a specific resource (file system, CMIS,...) create by configuration a resource provider app.

Unified Choose Dialog

In order to choose an asset (display an image/link into a page), we need to define a new choose dialog that allows to select an asset document/folder served by different sources (Internal like JCR, external CMIS server).

This choose dialog display assets in tree/list/thumbnail views and is based on a content app.

Choose dialog

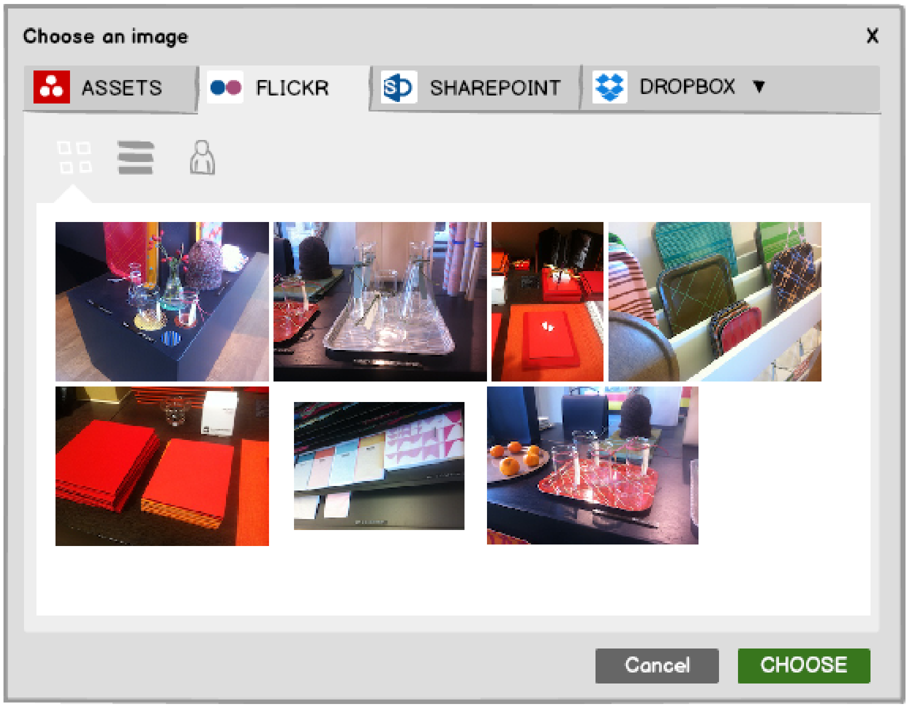


Name	Title	Template	Created
□ ▾ web	Home page	Home	2013-03-02
□ ▸ photography	Products	Section	2013-03-04
□ ▸ branding	Services	Section	2013-03-04
□ main_teaser.png	About us	Section	2013-03-04
□ campaign_mar1.tiff	Contact us	Section	2013-03-04

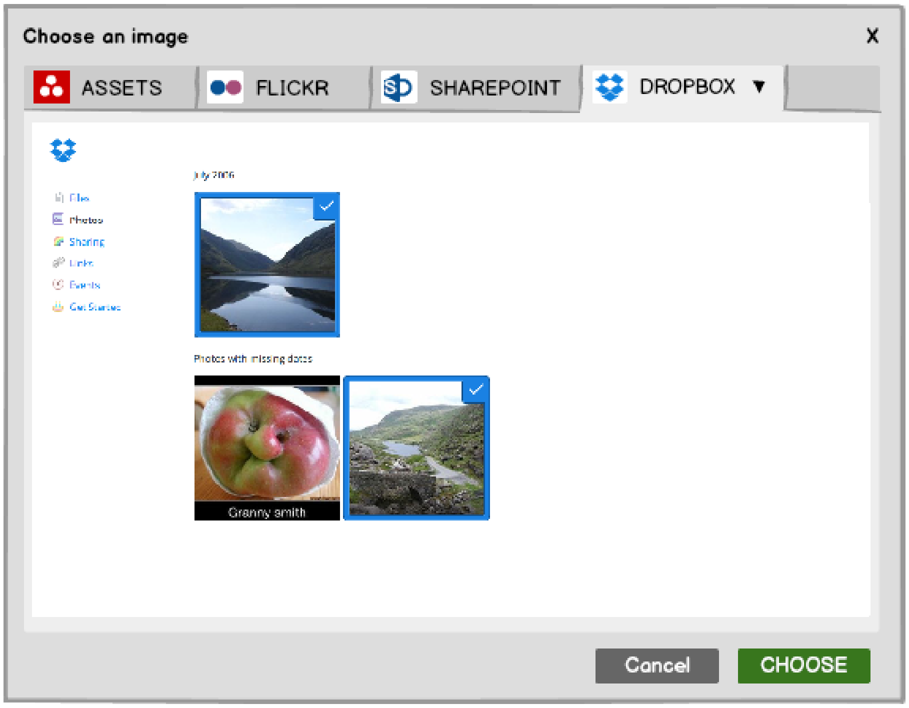
The Chooser dialog:

- uses the light dialog layout
- offers tabs for every asset app
- is capable of filtering apps and items shown by media type

In this example, we only show image asset sources here. For every source, we show folders and images, and filter out any other media type such as documents or videos.



The FLICKR app offers thumbs and list view, and adds another view to browse by account. It also doesn't offer any search.



The DROPBOX Photos integration is quite basic: it only shows a dropbox image picker inside an iframe.

The current displays limits the maximum amount of tabs to four. The triangle in the Dropbox tab allows to access additional, currently hidden data sources (to be implemented separately to apply to all tabs everywhere).

The choose dialog:

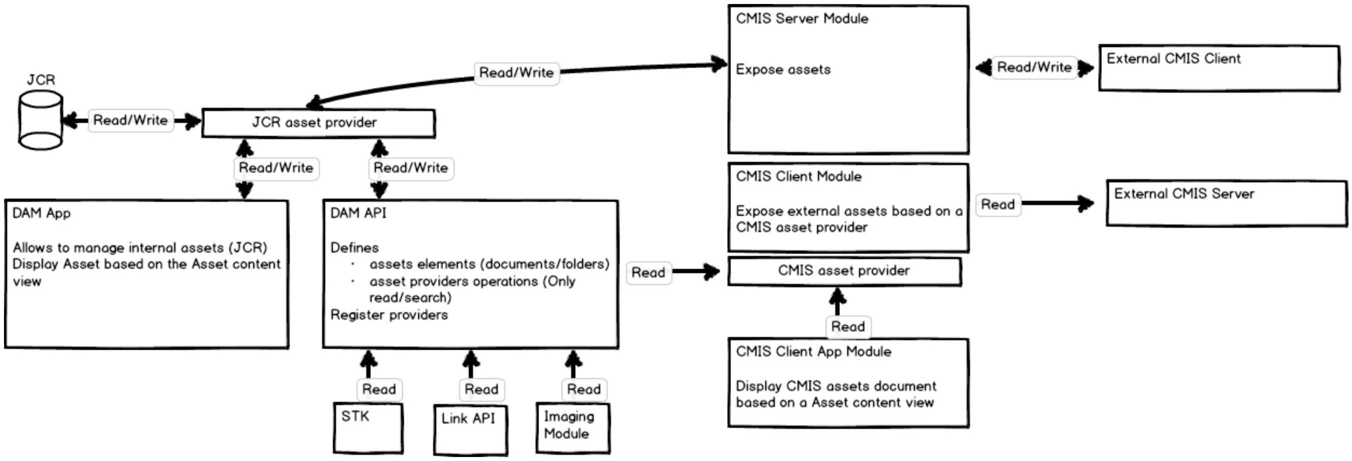
- Display (as tabs) the configured assets providers **choose dialog view**.
- Each provider is linked to:
  - dedicated content app (display assets)
  - specific actions (internal assets have an additional upload action)

Out of scope

- CRUD operations based on asset API. Only Read operations are defined and supported.
- Rewrite of the DAM App in order to use DAM API
- Federated view (App that display assets exposed by multiple providers)
- Federated search is out of scope for the moment.

Annexe

- first version of **the concept page**Concept - first iteration DAM 2.0 & CMIS
- Ideal target (most important: DAM App + CMIS Server Module use JCR asset provider!)


JCR
Read/Write
JCR asset provider
Read/Write
CMIS Server Module
Expose assets
Read/Write
External CMIS Client
Read/Write
Read/Write
DAM App
Allows to manage internal assets (JCR)
Display Asset based on the Asset content view
DAM API
Defines
· assets elements (documents/folders)
· asset providers operations (Only read/search)
Register providers
CMIS Client Module
Expose external assets based on a CMIS asset provider
Read
External CMIS Server
Read
CMIS asset provider
Read
CMIS Client App Module
Display CMIS assets document based on a Asset content view
Read
Read
Read
STK
Link API
Imaging Module