Embargo

What is an Embargo?

An embargo is a temporary access restriction placed on metadata or bitstreams. Its scope or duration may vary, but the fact that it eventually expires is what distinguishes it from other content restrictions. For example, it is not unusual for content destined for DSpace to come with permanent restrictions on use or access based on license-driven or other IP-based requirements that limit access to institutionally affiliated users. Restrictions such as these are imposed and managed using standard administrative tools in DSpace, typically by attaching specific policies to Items, Collections, Bitstreams, etc. Embargo functionality was originally introduced as part of DSpace 1.6, enabling embargoes on the level of items that applied to all bitstreams included in the item. In DSpace 3.0, this functionality has been extended for the XML User Interface, enabling embargoes on the level of individual bitstreams.

DSpace 3.0 XMLUI Embargo Functionality

Incompatible with JSPUI

By enabling the DSpace 3.0 Embargo functionality you will break JSPUI submission functionality. JSPUI is not yet compatible with the new XMLUI Embargo. If you use JSPUI, please see the Pre-DSpace 3.0 Embargo section.

Embargoes can be applied per *item* and per *bitstream*. The *item* level embargo will be the default for every *bitstream* , although it could be customized at *bitstream* level.

As a DSpace administrator, you can choose to integrate either Simple or Advanced dialog screens as part of the item submission process. These are outlined in detail in the sections Simple Embargo Settings and Advanced Embargo Settings.

This preference is stored in the dspace.cfg value xmlui.submission.restriction.enableAdvancedForm.

On the level of an individual item, a new Private/Public state has been introducted to control the visibility of item metadata in the different indexes serving the DSpace web interface (search, browse, discovery), as well as machine interfaces (REST-API, OAI-PMH, …)

The following functionality has been added in DSpace 3.0:

- *Browse private items*
- *Submission Process*
  - Simple/Advanced Access Step
  - Upload with embargo step
- *Edit Item*

    - Make it Private
    - Make it Public

The following functionality has been modified in DSpace 3.0:

- Edit Item
- Edit Bitstream
- Wildcard Policy Admin Tool

Configuring and using Embargoes in DSpace 3.0

Incompatible with JSPUI

By enabling the DSpace 3.0 Embargo functionality you will break JSPUI submission functionality. JSPUI is not yet compatible with the new XMLUI Embargo. If you use JSPUI, please see the Pre-DSpace 3.0 Embargo section.

Introduction

The following sections describe the steps needed to configure and use the new Embargo functionality in DSpace 3.0.
This embargo functionality is implemented using DSpace's system for ResourcePolicies.

Database

As a first step, the following script needs to be executed to ensure that your DSpace database gets extended with 3 new fields, required by the new embargo.

- dspace/etc/[postgres-oracle]/database_schema_18-3.sql

For PostgreSQL databases, the executed commands are:

```
ALTER TABLE resourcepolicy ADD rpname VARCHAR(30);
ALTER TABLE resourcepolicy ADD rptype VARCHAR(30);
ALTER TABLE resourcepolicy ADD rpdescription VARCHAR(100);
ALTER TABLE item ADD discoverable BOOLEAN;

update item set discoverable=true;
```

For Oracle databases, the executed commands are:

```
ALTER TABLE resourcepolicy
  ADD (
        rpname rpname VARCHAR2(30);
        rpdescription VARCHAR2(100);
        rptype VARCHAR2(30)
      );

ALTER TABLE item ADD discoverable NUMBER(1);
```

dspace.cfg

As already mentioned the user will be given the opportunity to choose between:

- Simple Embargo Settings
- Advanced Embargo Settings

To switch between the two, you need to set following variable in the *dspace.cfg* file.

```
xmlui.submission.restrictstep.enableAdvancedForm=false
```

Submission Process

item-submission.xml

To enable the new embargo changes are required to the *item-submission.xml* file, located in your config directory. This file determines which steps are executed in the submission of a new item.

Two new submission steps have been introduced in the file. By default, they are not activated yet:

- *AccessStep*: the step in which the user can set the embargo at item level, effectively restricting access to the item metadata.
- *UploadWithEmbargoStep*: the step in which the user can set the embargo at bitstream level. **If this step is enabled, the old *UploadStep* must be disabled, leaving both steps enabled will result in a system failure.**

Here is an extract from the new file:

```
<!--Step 3 will be to Manage Item access.
       <step>
           <heading>submit.progressbar.access</heading>
           <processing-class>org.dspace.submit.step.AccessStep</processing-class>
           <xmlui-binding>org.dspace.app.xmlui.aspect.submission.submit.AccessStep</xmlui-binding>
           <workflow-editable>true</workflow-editable>
       </step>
       -->

<!-- Step 4 Upload Item with Embargo Features (not supported in JSPUI)
            to enable this step, please make sure to comment-out the previous step "UploadStep"
       <step>
           <heading>submit.progressbar.upload</heading>
           <processing-class>org.dspace.submit.step.UploadWithEmbargoStep</processing-class>
           <xmlui-binding>org.dspace.app.xmlui.aspect.submission.submit.UploadWithEmbargoStep</xmlui-binding>
           <workflow-editable>true</workflow-editable>
       </step>
        -->
```

To enable the new Embargo, ensure that the new steps are uncommented, while the old UploadStep needs to be commented out.

Simple Embargo Settings

Using the simple embargo settings, submitters will be able to define embargoes bound to specific dates that are applied to all anonymous and default read access. To keep the interface simple, options to apply embargoes for particular groups of DSpace users are not shown. The simple embargo settings interface assumes that embargoes always start immediately upon submission, so only end dates are configurable.

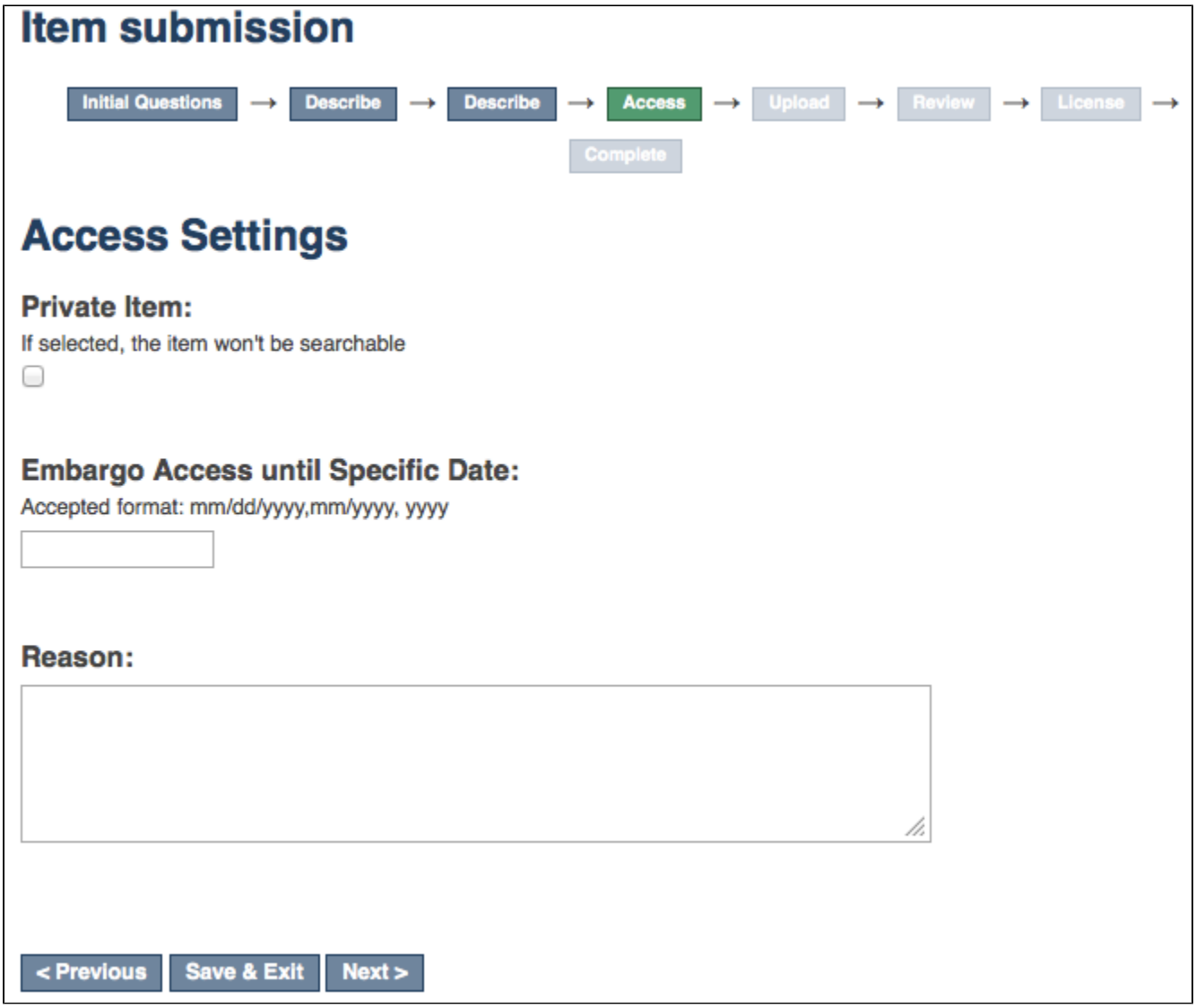
AccessStep

The simple *AccessStep* Embargo form renders three options for the user:

- *Private item*: to hide an item's metadata from all search and browse indexes, as well as external interfaces such as OAI-PMH.
- *Embargo Access until Specific Date*: to indicate a date until when the item will be embargoed.
- *Reason*: to elaborate on the specific reason why an item is under embargo.

When Embargo is set it applies to Anonymous or to any other Group that is indicated to have *default read access* for that specific collection.

This shows how the Access step is rendered, using the simple embargo settings:



UploadWithEmbargoStep

The simple *UploadWithEmbargoStep* form renders two new fields for the user:

- *Embargo Access until Specific Date*: to indicate a date until when the *bitstream* will be embargoed. When left empty, no embargo will be applied.
- *Reason*: to elaborate on the specific reason why the *bitstream* is under embargo.

These fields will be preloaded with the values set in the *AccessStep.*

The following picture shows the form for the Upload step, rendered using the simple embargo settings with preloaded values:

Advanced Embargo Settings

The Advanced Embargo settings are really designed with a submitter in mind who is aware of user groups in DSpace and has notions of the way how Policies work.

AccessStep

The Advanced *AccessStep* Embargo step allows the users to manage more fine-grained resource policies to attach to the item.

The form will render the following fields:

- *Policies List*: list of the custom policies already added
- *Private Item*: to hide an item's metadata from all search and browse indexes, as well as external interfaces such as OAI-PMH.
- *Name*: to give a name to the policy
- *Groups*: to indicate the group for which the policy will apply to.
- *Visible/Embargoed*: to set if the Item will be visible or embargoed for that specific group
- *Embargo Access until Specific Date*: to indicate a date until the item will be embargoed
- *Reason*: to elaborate on the specific reason why the policy is applied.

Last two fields will be enabled only in case of *Embargoed* has been selected.

This step gives the opportunity to the user to manage the policy manually, so that combination as the following will be possible:

- Set Embargo for Anonymous
- Set Embargo for anyone, except for the users belonging to a specific group
- Set Embargo for specific groups, but not for another groups ...

Here is a screenshot of the Access step form that will be rendered for the advanced embargo settings:

Item submission

Initial Questions → Describe → Describe → Access → Upload → Review → License → Complete

Policies List

Name	Action	Group	Start Date	End Date		
Embargo Policy	READ	Anonymous	2013-01-01		Edit	Remove

Access Settings

Private Item:

If selected, the item won't be searchable

☐

Name:

Groups:

Administrator

Visible/Embargoed:

◉ Item will be visibile once accepted into archive ○ Embargo Access until Specific Date

Accepted format: mm/dd/yyyy,mm/yyyy, yyyy

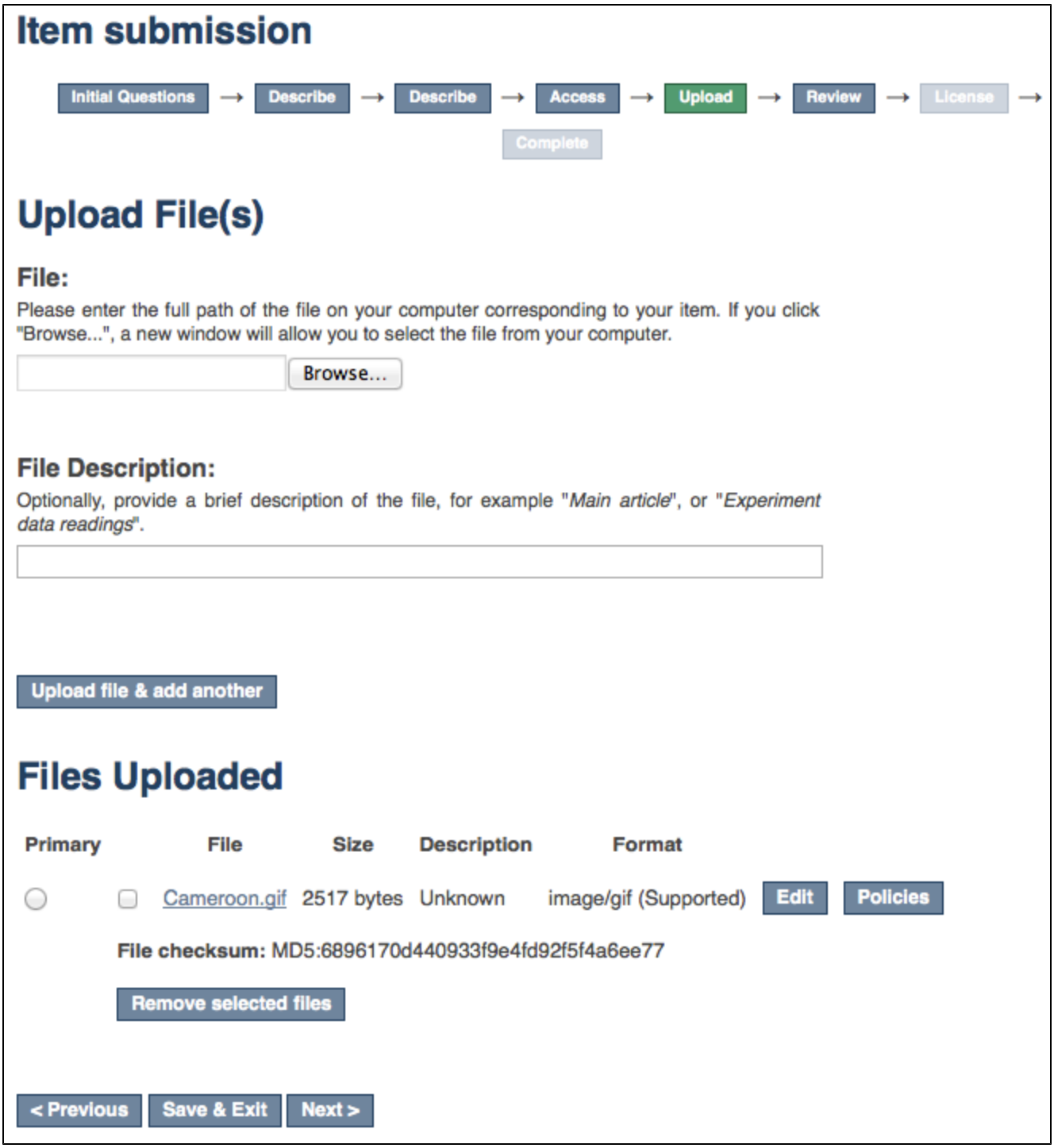
Reason:

Confirm Policy & add another

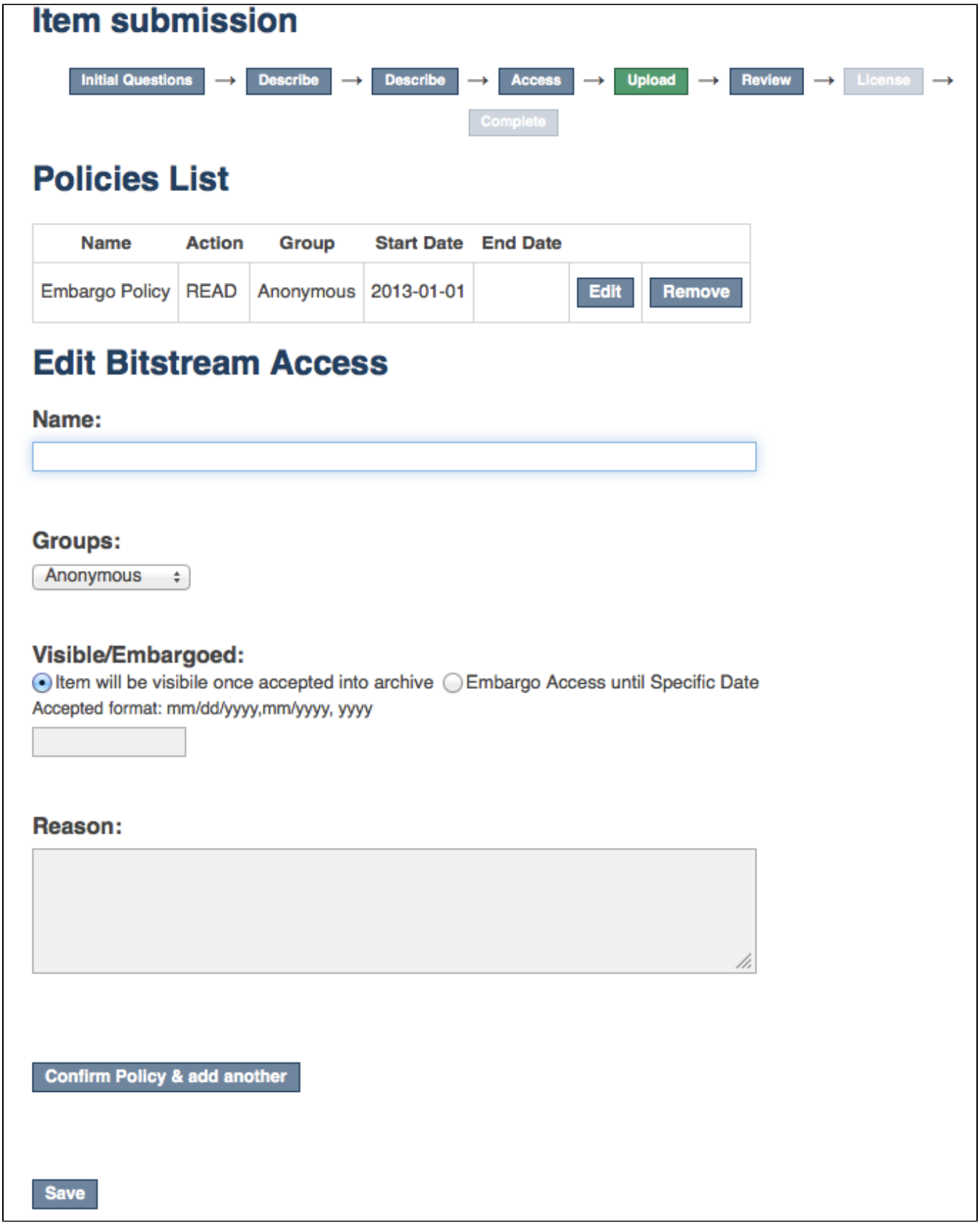
< Previous | Save & Exit | Next >

UploadWithEmbargoStep

UploadWithEmbargoStep for Advanced Embargo settings displays an additional *Policies* button next to *Edit* in the list of uploaded files. Clicking it brings you to the a page where you can edit existing policies on the bitstream and add new ones.



When the button will be pushed a form similar to the one in the *AccessStep* will be render and makes it possible to manage the policies at *bitstream* level.



Restrict list of displayed groups to specific (sub)groups

For large instances of DSpace, the list of Groups can be quite long. Groups can be nested. This means that not only EPersons can be members of groups, but groups themselves can belong to other groups.

When advanced embargo settings are enabled, you can limit the list of groups displayed to the submitters to subgroups of a particular group.

To use this feature, assign the super group name to following configuration value in dspace.cfg:

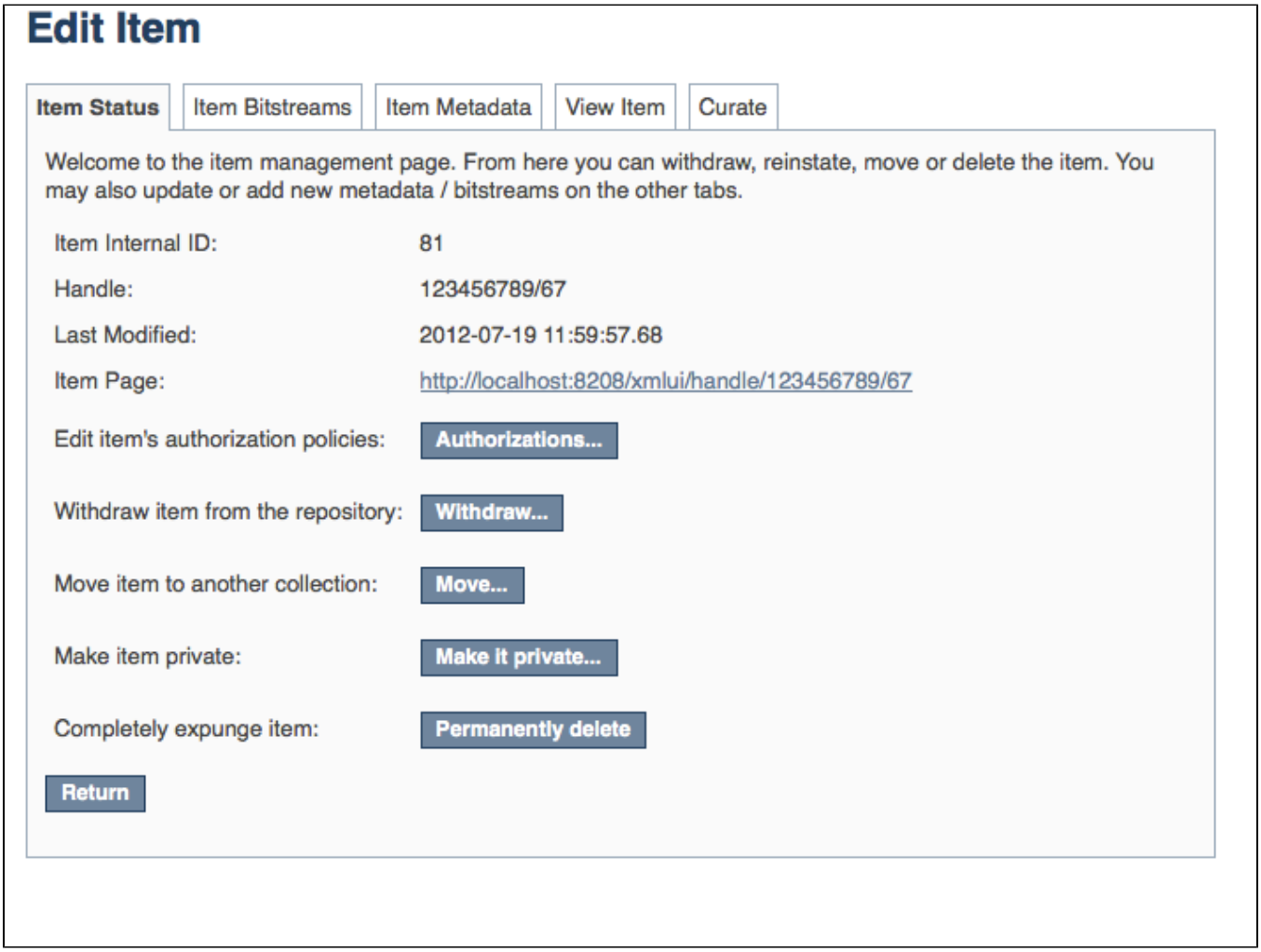
xmlui.submission.restrictstep.groups=name_of_the_supergroup

Once a specific group is configured as supergroup here, only the groups belonging to the indicated group will be loaded in the selection dialogs. By default, all groups are loaded.

Private/Public Item

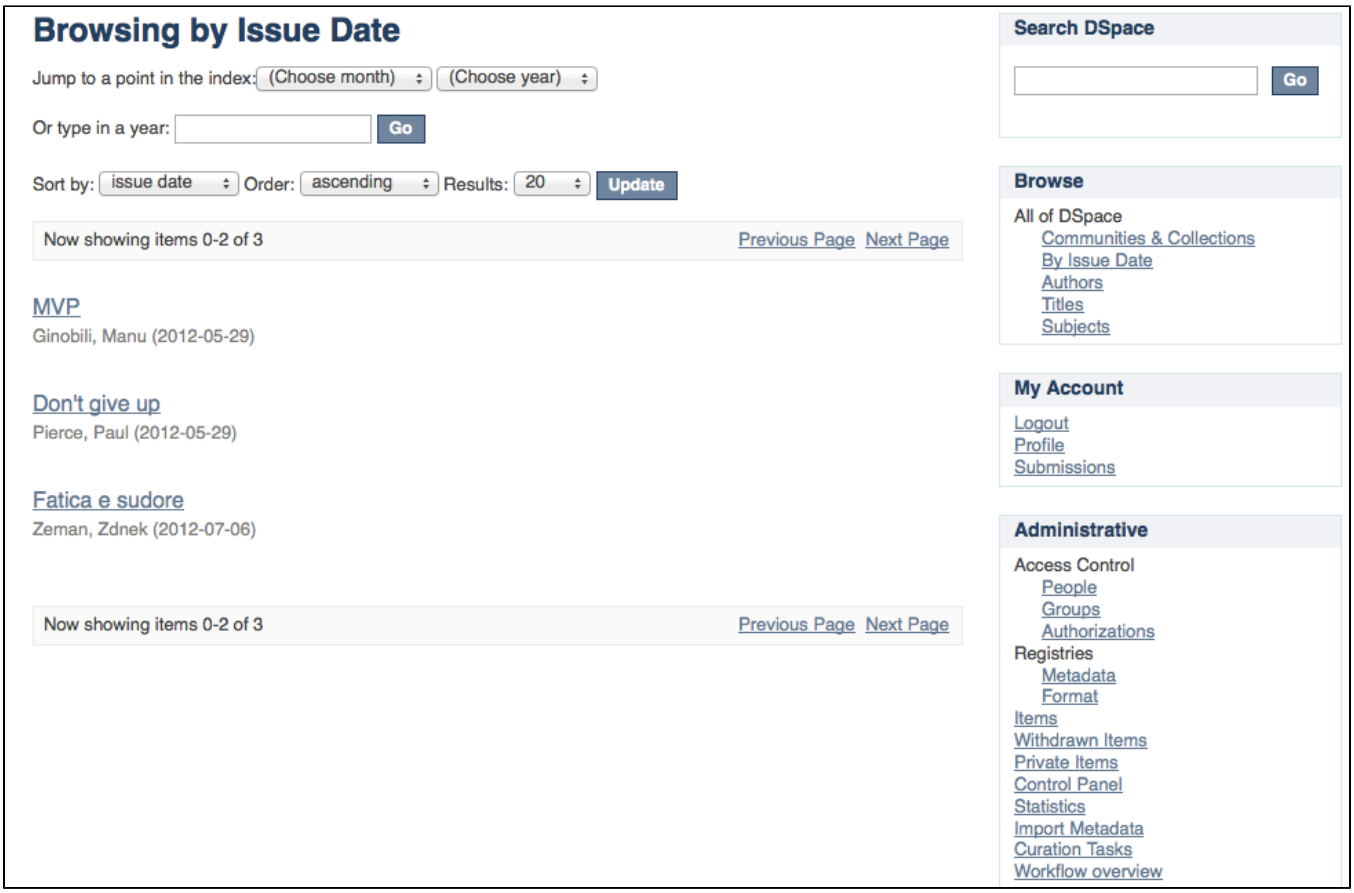
It is possible to adjust the Private/Public state of an item also after it has been archived in the repository.

Here is a screenshot showing the updated *Edit Item* dialog:



Private items are not retrievable through the DSpace search, browse or Discovery indexes.

Therefor, an admin-only view has been created to browse all private items. Here is a screenshot of this new form:



Pre-3.0 Embargo Migration Routine

A migration routine has been developed to migrate the current Embargo to the new one.

To execute it occurs to run the following command:

```
./dspace migrate-embargo -a
```

Technical Specifications

Introduction

The following sections illustrate the technical changes that have been made to the *back-end* to add the new *Advanced Embargo* functionality.

ResourcePolicy

When the embargo will be set at *item* level or *bitstream* level a new *ResourcePolicy* will be added.

Three new attributes have been introduced in the *ResourcePolicy* class:

- *rpname*: resource policy name
- *rptype*: resource policy type
- *rpdescription*: resource policy description

While *rpname* and *rpdescription _are fields manageable by the users the _rptype* is a fields managed by the system. It represents a type that a resource policy can assume beteween the following:

- TYPE_SUBMISSION: all the policies added automatically during the submission process
- TYPE_WORKFLOW: all the policies added automatically during the workflow stage
- TYPE_CUSTOM: all the custom policies added by users
- TYPE_INHERITED: all the policies inherited by the *DSO* father.

Here is an example of all information contained in a single policy record:

```
policy_id: 4847
resource_type_id: 2
resource_id: 89
action_id: 0
eperson_id:
epersongroup_id: 0
start_date: 2013-01-01
end_date:
rpname: Embargo Policy
rpdescription:  Embargoed through 2012
rptype: TYPE_CUSTOM
```

Item

To manage **Private/Public** state a new *boolean* attribute has been added to the Item:

- isDiscoverable

When and Item is private the attribute will assume the value ***false***.

Item.inheritCollectionDefaultPolicies(Collection c)

This method has been adjusted to leave custom policies, added by the users, in place and add the default collection policies only if there are no custom policies.

AuthorizeManager

Some methods have been changed on *AuthorizeManager _to manage* _the new fields ans some convenient methods have been introduced like:

```
public static List<ResourcePolicy> findPoliciesByDSOAndType(Context c, DSpaceObject o, String type);
public static void removeAllPoliciesByDSOAndTypeNotEqualsTo(Context c, DSpaceObject o, String type);
public static boolean isAnIdenticalPolicyAlreadyInPlace(Context c, DSpaceObject o, ResourcePolicy rp);
public static ResourcePolicy createOrModifyPolicy(ResourcePolicy policy, Context context, String name, int
idGroup, EPerson ePerson, Date embargoDate, int action, String reason, DSpaceObject dso);
```

Withdraw Item

The feature to withdraw an item from the repository has been modified to keep all the custom policies in place.

Reinstate Item

The feature to reinstate an item in the repository has been modified to preserve existing custom policies.

Pre-DSpace 3.0 Embargo Compatibility

The Pre-DSpace 3.0 embargo functionality (see below) has been modified to adjust the policies setter and lifter. These classes now also set the dates within the policy objects themselves in addition to setting the date in the item metadata.

Pre-DSpace 3.0 Embargo

Embargo model and life-cycle

Functionally, the embargo system allows you to attach 'terms' to an item before it is placed into the repository, which express how the embargo should be applied. What do 'we mean by terms' here? They are really any expression that the system is capable of turning into (1) the time the embargo expires, and (2) a concrete set of access restrictions. Some examples:

"2020-09-12" - an absolute date (i.e. the date embargo will be lifted)
"6 months" - a time relative to when the item is accessioned
"forever" - an indefinite, or open-ended embargo
"local only until 2015" - both a time and an exception (public has no access until 2015, local users OK immediately)
"Nature Publishing Group standard" - look-up to a policy somewhere (typically 6 months)

These terms are 'interpreted' by the embargo system to yield a specific date on which the embargo can be removed or 'lifted'., and a specific set of access policies. Obviously, some terms are easier to interpret than others (the absolute date really requires none at all), and the 'default' embargo logic understands only the most basic terms (the first and third examples above). But as we will see below, the embargo system provides you with the ability to add in your own 'interpreters' to cope with any terms expressions you wish to have. This date that is the result of the interpretation is stored with the item and the embargo system detects when that date has passed, and removes the embargo ("lifts it"), so the item bitstreams become available. Here is a more detailed life-cycle for an embargoed item:

Terms assignment

The first step in placing an embargo on an item is to attach (assign) 'terms' to it.
If these terms are missing, no embargo will be imposed. As we will see below, terms are carried in a configurable DSpace metadata field, so assigning terms just means assigning a value to a metadata field. This can be done in a web submission user interface form, in a SWORD deposit package, a batch import, etc. - anywhere metadata is passed to DSpace. The terms are not immediately acted upon, and may be revised, corrected, removed, etc, up until the next stage of the life-cycle. Thus a submitter could enter one value, and a collection editor replace it, and only the last value will be used. Since metadata fields are multivalued, theoretically there can be multiple terms values, but in the default implementation only one is recognized.

Terms interpretation/imposition

In DSpace terminology, when an Item has exited the last of any workflow steps (or if none have been defined for it), it is said to be 'installed' into the repository. At this precise time, the 'interpretation' of the terms occurs, and a computed 'lift date' is assigned, which like the terms is recorded in a configurable metadata field. It is important to understand that this interpretation happens only once, (just like the installation), and cannot be revisited later. Thus, although an administrator can assign a new value to the metadata field holding the terms after the item has been installed, this will have no effect on the embargo, whose 'force' now resides entirely in the 'lift date' value. For this reason, you cannot embargo content already in your repository (at least using standard tools). The other action taken at installation time is the actual imposition of the embargo. The default behavior here is simply to remove the read policies on all the bundles and bitstreams except for the "LICENSE" or "METADATA" bundles. See section V. below for how to alter this behavior. Also note that since these policy changes occur before installation, there is no time during which embargoed content is 'exposed' (accessible by non-administrators). The terms interpretation and imposition together are called 'setting' the embargo, and the component that performs them both is called the embargo 'setter'.

Embargo period

After an embargoed item has been installed, the policy restrictions remain in effect until removed. This is not an automatic process, however: a 'lifter' must be run periodically to look for items whose 'lift date' is past. Note that this means the effective removal of an embargo is **not** the lift date, but the earliest date after the lift date that the lifter is run. Typically, a nightly cron-scheduled invocation of the lifter is more than adequate, given the granularity of embargo terms. Also note that during the embargo period, all metadata of the item remains visible.This default behavior can be changed. One final point to note is that the 'lift date', although it was computed and assigned during the previous stage, is in the end a regular metadata field. That means, if there are extraordinary circumstances that require an administrator (or collection editor - anyone with edit permissions on metadata) to change the lift date, they can do so. Thus, they can 'revise' the lift date without reference to the original terms. This date will be checked the next
time the 'lifter' is run. One could immediately lift the embargo by setting the lift date to the current day, or change it to 'forever' to indefinitely postpone lifting.

Embargo lift

When the lifter discovers an item whose lift date is in the past, it removes (lifts) the embargo. The default behavior of the lifter is to add the resource policies **that would have been added** had the embargo not been imposed. That is, it replicates the standard DSpace behavior, in which an item inherits it's policies from its owning collection. As with all other parts of the embargo system, you may replace or extend the default behavior of the lifter (see section V. below). You may wish, e.g. to send an email to an administrator or other interested parties, when an embargoed item becomes available.

Post embargo

After the embargo has been lifted, the item ceases to respond to any of the embargo life-cycle events. The values of the metadata fields reflect essentially historical or provenance values. With the exception of the additional metadata fields, they are indistinguishable from items that were never subject to embargo.

Configuration

DSpace embargoes utilize standard metadata fields to hold both the "terms" and the "lift date". Which fields you use are configurable, and no specific metadata element is dedicated or pre-defined for use in embargo. Rather, you specify exactly what field you want the embargo system to examine when it needs to find the terms or assign the lift date.

The properties that specify these assignments live in dspace.cfg:

```
# DC metadata field to hold the user-supplied embargo terms
embargo.field.terms = SCHEMA.ELEMENT.QUALIFIER

# DC metadata field to hold computed "lift date" of embargo
embargo.field.lift = SCHEMA.ELEMENT.QUALIFIER
```

You replace the placeholder values with real metadata field names. If you only need the "default" embargo behavior - which essentially accepts only absolute dates as "terms",
this is the only configuration required, except as noted below.

There is also a property for the special date of 'forever':

```
# string in terms field to indicate indefinite embargo
embargo.terms.open = forever
```

which you may change to suit linguistic or other preference.

You are free to use existing metadata fields, or create new fields. If you choose the latter, you must understand that the embargo system does **not** create or configure these fields: i.e. you must follow all the standard documented procedures for actually creating them (i.e. adding them to the metadata registry, or to display templates, etc) - this does not happen automatically. Likewise, if you want the field for "terms" to appear in submission screens and workflows, you must follow the documented procedure for configurable submission (basically, this means adding the field to input-forms.xml). The flexibility of metadata configuration makes if easy for you to restrict embargoes to specific collections, since configurable submission can be defined per collection.

Key recommendations:

1. Use a local metadata schema. Breaking compliance with the standard Dublin Core in the default metadata registry can create a problem for the portability of data to/from of your repository.
2. If using existing metadata fields, avoid any that are automatically managed by DSpace. For example, fields like "date.issued" or "date.accessioned" are normally automatically assigned, and thus must not be recruited for embargo use.
3. Do not place the field for "lift date" in submission screens. This can potentially confuse submitters because they may feel that they can directly assign values to it. As noted in the life-cycle above, this is erroneous: the lift date gets assigned by the embargo system based on the terms. Any pre-existing value will be over-written. But see next recommendation for an exception.
4. As the life-cycle discussion above makes clear, after the terms are applied, that field is no longer actionable in the embargo system. Conversely, the "lift date" field is not actionable **until** the application. Thus you may want to consider configuring both the "terms" and "lift date" to use the same metadata field. In this way, during workflow you would see only the terms, and after item installation, only the lift date. If you wish the metadata to retain the terms for any resaon, use 2 distinct fields instead.

Operation

After the fields defined for terms and lift date have been assigned in dspace.cfg, and created and configured wherever they will be used, you can begin to embargo items simply by entering data (dates, if using the default setter) in the terms field. They will automatically be embargoed as they exit workflow. For the embargo to be lifted on any item, however, a new administrative procedure must be added: the "embargo lifter" must be invoked on a regular basis. This task examines all embargoed items, and if their "lift date" has passed, it removes the access restrictions on the item. Good practice dictates automating this procedure using cron jobs or the like, rather than manually running it.
The lifter is available as a target of the 1.6 DSpace launcher - see launcher documentation for details.

Extending embargo functionality

The 1.6 embargo system supplies a default "interpreter/imposition" class (the "Setter") as well as a "Lifter", but they are fairly rudimentary in several respects.

Setter

The default setter recognizes only two expressions of terms: either a literal, non-relative date in the fixed format "yyyy-mm-dd" (known as ISO 8601), or a special string used for open-ended embargo (the default configured value for this is 'forever', but this can be changed in dspace.cfg to 'toujours', 'unendlich', etc). It will perform a minimal sanity check that the date is not in the past. Similarly, the default setter will only remove all read policies as noted above, rather than applying more nuanced rules (e.g allow access to certain IP groups, deny the rest). Fortunately, the setter class itself is configurable and you can 'plug in' any behavior you like, provided it is written in java and conforms to the setter interface. The dspace.cfg property:

```
# implementation of embargo setter plugin - replace with local implementation if applicable
plugin.single.org.dspace.embargo.EmbargoSetter = org.dspace.embargo.DefaultEmbargoSetter
```

controls which setter to use.

Lifter

The default lifter behavior as described above - essentially applying the collection policy rules to the item - might also not be sufficient for all purposes. It also can be replaced with another class:

```
implementation of embargo lifter plugin - - replace with local implementation if applicable
plugin.single.org.dspace.embargo.EmbargoLifter = org.dspace.embargo.DefaultEmbargoLifter
```