

Embargo

- 1 [What is an Embargo?](#)
- 2 [DSpace Embargo Functionality](#)
 - 2.1 [Managing Embargoes on existing Items](#)
- 3 [Configuring and using Embargo in DSpace Submission User Interface](#)
 - 3.1 [Enabling Item-level Embargo](#)
 - 3.2 [Configuring Embargo / Access Restriction options](#)
 - 3.3 [Private/Public \(or Non-Discoverable/Discoverable\) Item](#)
 - 3.4 [Pre-3.0 Embargo Migration Routine](#)
- 4 [Technical Specifications](#)
 - 4.1 [Introduction](#)
 - 4.2 [ResourcePolicy](#)
 - 4.3 [Item](#)
 - 4.4 [Item.inheritCollectionDefaultPolicies\(Collection c\)](#)
 - 4.5 [AuthorizeService](#)
 - 4.6 [Withdraw Item](#)
 - 4.7 [Reinstate Item](#)
 - 4.8 [Pre-DSpace 3.0 Embargo Compatibility](#)
- 5 [Creating Embargoes via Metadata](#)
 - 5.1 [Introduction](#)
 - 5.2 [Setting Embargo terms via metadata](#)
 - 5.2.1 [Terms assignment](#)
 - 5.2.2 [Terms interpretation/imposition](#)
 - 5.2.3 [Embargo period](#)
 - 5.3 [Configuration of metadata fields](#)
 - 5.4 [Operation](#)
 - 5.5 [Extending embargo functionality](#)
 - 5.5.1 [Setter](#)
 - 5.5.2 [Lifter](#)

What is an Embargo?

An embargo is a temporary access restriction placed on metadata or bitstreams (i.e. files). 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 access policies (aka "resource 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. Since DSpace 3.0, this functionality has been extended to the Submission User Interface, enabling embargoes on the level of individual bitstreams.

DSpace Embargo Functionality

Embargoes can be applied per *item* (including metadata) and per *bitstream* (i.e. file). The *item* level embargo will be the default for every *bitstream*, although it could be customized at *bitstream* level.

When an embargo is set on either an item level or a bitstream level, a new ResourcePolicy (i.e. access policy) is added to the corresponding Item or Bitstream. **This ResourcePolicy will automatically control the lifting of the embargo (when the embargo date passes).** An embargo lift date is generally stored as the "start date" of such a policy. Essentially, this means that the access rights defined in the policy do not get applied until *after that date passes* (and prior to that date, the access rights will default to Admin only).

The scheduled, manual "embargo-lifter" commands (used prior to DSpace 3) are no longer necessary and not recommended to run.

Managing Embargoes on existing Items


Administrators are able to change the lift date of any embargo by editing the authorization policy (ResourcePolicy) on the object. These authorization policies can be managed from the Edit Item screen by clicking on "Authorizations".

- To add an embargo, edit the appropriate policy and set a "start date". To add an full Item embargo (including metadata), edit the Item policy. To embargo individual bitstreams, edit the appropriate Bitstream policy.
- To remove an embargo, edit the appropriate policy, and clear out the "start date".
- To change an embargo, edit the appropriate policy, and change the "start date" to a new date.

Changes to the embargo should take effect immediately. However, as Administrators have full access to embargoed items, you may need to log out first. After logging out, you will be subject to the embargo.

Configuring and using Embargo in DSpace Submission User Interface

Available in DSpace 7.2 and above

 In DSpace 7.2 and above, both Item-level embargoes and bitstream (file) level embargoes are supported in the Submission user interface. In DSpace 7.1 and 7.0, the Submission user interface only supported embargoes on specified bitstreams (files). However, item-level embargoes could be added after submissions were accepted using the "Manage Embargoes on existing Items" approach described above.

Enabling Item-level Embargo

While Bitstream-level embargoes are enabled by default, Item-level embargoes currently are not. However, enabling them is easy. Simply update your `item-submission.xml` to include this tag in your `<submission-process>`:

```
<submission-process name="traditional">
  ...

  <!-- This step enables embargoes and other access restrictions at the Item level -->
  <step id="itemAccessConditions"/>
</submission-process>
```

After making this update, you will need to restart your backend (REST API) for the changes to take effect.

Configuring Embargo / Access Restriction options

Starting in DSpace 7, embargo (and lease) settings are configurable via a Spring Bean configuration file `[dspace]/config/spring/api/access-conditions.xml`

For detailed information on configuring your Embargo options (and other related options like lease or restrict to a particular group of users), see the [Submission User Interface](#) documentation. Specifically these two sections:

- For Bitstream embargo / access options, see the section on "Configuring the File Upload step" of the [Submission User Interface](#)
- For Item embargo / access options, see the section on "Configuring the Item Access Conditions step" of the [Submission User Interface](#)

Private/Public (or Non-Discoverable/Discoverable) Item

It is also possible to adjust the Private/Public (or Non-Discoverable/Discoverable) state of an item after it has been archived in the repository. This can be achieved from either the "Admin Search" (`/admin/search`), or from the "Status" tab under "Edit Item".

Private (or non-Discoverable) items are not retrievable through the DSpace search, browse or Discovery indexes. However, they are accessible via a direct link. It is possible to create a publicly accessible, non-discoverable item...in which case it can only be shared via a direct link. But, once anyone has that link, it is available anonymously.

Therefore, an "Admin Search" option is provided, which allows you to search across all items, including private or withdrawn items. You can also filter your results to display only private items.

Pre-3.0 Embargo Migration Routine

If you have just upgraded from a DSpace 1.x.x version, any embargoes that are currently "in effect" will need to be migrated into ResourcePolicies. Prior to 3.0, embargoes in DSpace were managed entirely in metadata fields (and required running a scheduled "embargo-lifter" command). However, DSpace now stores all embargo information directly on ResourcePolicies (i.e. "access policies"). These ResourcePolicies automatically "lift" an embargo after the embargo date passes.

In order to migrate old embargoes into ResourcePolicies, a migration routine has been developed. **Please note that this migration routine should only need to be run ONCE** (immediately after an upgrade from 1.x.x to a more recent version of DSpace). After that point, any newly defined embargoes will automatically be stored on ResourcePolicies.

To execute it, run the following command:

```
[dspace]/bin/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 an embargo is 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 users, the *rptype* is managed by DSpace itself. It represents a type that a resource policy can assume, among 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 from the enclosing object (for Item, a Collection; for Bitstream, an Item).

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 an 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.

AuthorizeService

Some methods have been changed on *AuthorizeService* to manage the new fields and some convenience methods have been introduced:

```
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.

Creating Embargoes via Metadata

Introduction

Prior to DSpace 3.0, all DSpace embargoes were stored as metadata. While embargoes are no longer stored permanently in metadata fields (they are now stored on ResourcePolicies, i.e. access policies), embargoes *can still be initialized via metadata fields*.

This ability to create/initialize embargoes via metadata is extremely powerful if you wish to submit embargoed content via electronic means (such as [Importing Items via Simple Archive Format](#), [SWORDv1](#), [SWORDv2](#), etc).

Setting Embargo terms via metadata

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 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. 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, and recorded as part of the ResourcePolicy (aka policy) of the Item. Once the lift date has been assigned to the ResourcePolicy, the metadata field which defined the embargo is **cleared**. From that point forward, all embargo information is controlled/defined by the ResourcePolicy.

It is important to understand that this interpretation happens only once, (just like the installation). Therefore, **updating/changing an embargo cannot be done via metadata fields**. Instead, all embargo updates must be made to the ResourcePolicies themselves (e.g. ResourcePolicies can be managed from the Admin UI in the Edit Item screens).

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 the embargo date passes. Once the embargo date passes, the policy restrictions are automatically lifted. An embargo lift date is generally stored as the "start date" of a policy. Essentially, this means that the policy does not get applied until after that date passes (and prior to that date, the object defaults to Admin only access).

Administrators are able to change the lift date of the embargo by editing the policy (ResourcePolicy). These policies can be managed from the Edit Item screens.

Configuration of metadata fields

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 must 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 submission-forms.xml). The flexibility of metadata configuration makes it 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 reason, 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, and that the computed lift date will be stored on the ResourcePolicy

Extending embargo functionality

The embargo system supplies a default "interpreter/imposition" class (the "Setter") .

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

DEPRECATED: The Lifter is no longer used in the DSpace API, and is not recommended to utilize. Embargo lift dates are now stored on ResourcePolicies and, as such, are "lifted" automatically when the embargo date passes. Manually running a "lifter" may bypass this automatic functionality and result in unexpected results.

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
```