Configuration changes in the submission process

Outdated

These Docs are now outdated and have been moved to the official docs at Submission User Interface. This page is being kept temporarily while we verify all notes have been moved. Please reference the official docs instead of this page.

Before DSpace 7 two main files were used to configure the submission process:

- `item-submission.xml`
- `input-forms.xml`

There were also some configurations scattered through the `config/dspace.cfg` file.

With the switch to DSpace 7 we have decided to revise the concept behind the submission process. As we want to provide a RESTful application and a single-page UI experience (Angular), we don't want to enforce any more the concept of multi-steps wizard in the submission. For such reason we are replacing "Steps" and "Pages" in the above files with a more abstract concept of "section" that will be rendered by the UIs built on top of the REST API in various ways: as panels in the default proposal for the angular UI, or maybe as tabs or subsequent pages in custom UIs.

Moreover, we will try to rationalize the additional configurations, migrating them to self-contained files.

The `config/item-submission.xml`

The high-level structure is unchanged. The main differences are:

- each step **MUST** be defined in the step-definitions section, i.e. it is not anymore possible to define a step inline with the submission-definitions (see below).
- each step **MUST** have an unique ID (no more unnamed steps).
- each step **always** represents a single section, so if previously you had multiple pages to collect metadata, now you have different steps, one for each old page.
- an attribute `mandatory=[true|false]` has been introduced on the `step` element. When false, the section must be activated explicitly by the user by mean of an action on the UI or supplying data of interest of the section.
- the workflow-editable element has been replaced with a scope element to give more flexibility and make the configuration options for the sections the same available for the single metadata in the `input-form.xml` (now `submission-form.xml`).

As each page of the old input-forms now has become a separate "step" in the `item-submission.xml`, the `item-submission.xml` file now manages which metadata are available when a submission is done in a specific collection via the submission-map.

The `config/input-forms.xml` (now `config/submission-forms.xml`)

To reflect the big changes in this file's structure and purpose we have renamed the file to `submission-forms.xml`.

At the highest level the changes are:

- the form-map element is **not** anymore available. The mapping between collection and sequence of forms is now maintained in a single place: `config/item-submission.xml`.
- the form > page element is **not** anymore available. Each form consists of a single page. As said above, pages are grouped together in the `config/item-submission.xml`.
- a new child mandatory element `row` has been introduced in the form-definitions to allow to put more fields in a single row. The layout of the row is automatically defined by the number of fields in the row (i.e. 2 fields will mean a "col-6" bootstrap style) but can be customized using the new `style` element in the `field` tag.

The `value-pairs` element now automatically defines authorities when the value-pair is referenced by a form > fields > field > input-type **without** the need to manually register the authority in the dspace.cfg.

By default a new form named `bitstream-metadata` has been introduced. It configures the metadata requested for a bitstream during the submission. See the section about the new `access-conditions.xml` file below.

The configuration previously in `config/dspace.cfg`

The following configurations have been moved to the new `access-conditions.xml`. See the corresponding section.

- webui.submission.restrictstep.groups: see Embargo#Restrictlistofdisplayedgroupstospecific(sub)groups
- webui.submission.restrictstep.enableAdvancedForm
- webui.submit.upload.html5
- upload.max
- webui.submit.upload.required

NEW: `config/spring/api/access-conditions.xml`

With the PR https://github.com/DSpace/DSpace/pull/1889 a new configuration file will be introduced. It is a Spring bean configuration file named access-conditions.xml. This is used to configure the options to present in the upload section. Such configuration is exposed over the new REST API in the /config /submissionuploads endpoint.

The file has the following structure:

- the bean named *uploadConfigurationService* maps section configuration names (the section name used in the `item-submission.xml` file) to various upload configurations (UploadConfiguration beans).
- one or more UploadConfiguration beans. The *uploadConfigurationDefault* is an example of configuration where all the existent capabilities of DSpace < 7 are presented.

Each upload configuration allows to configure which *metadata* are requested to describe the bitstreams using the *name* assigned to the specific form configuration in the submission-form.xml. This mean that it is now possible to take full advantage of the metadata support at the bitstream level. A *maxSize* and *required* property can be also configured to specify the limit in bytes for file upload and if at least one file is required or not.

Moreover, it defines the list of acceptable access policies using the `options` list property, supporting the functionalities previously provided by the *UploadWithEmbargoStep*. This is a list of AccessConditionOption beans, each describing a ResourcePolicy which will be created during submission.

An empty list or null value for the options attribute means that the upload step doesn't allow the user to set a policy for the file. The file will get only the policies inherited from the collection according to the previous behavior of the *UploadStep*.

An AccessConditionOption is defined by the following properties:

- name: uniquely identify the policy template. It is stored in the ResourcePolicy.name once applied.
- groupName or selectGroupName: the first is used to bind the ResourcePolicy to a specific group; the latter is used to allow the selection of one of the subgroups in a group as principal of the created resourcepolicy. This is the equivalent of the previous Embargo#Restrictlistofdisplayedgroupstospecific(sub)groups configuration option.
- hasStartDate [true|false]. If true the policy to create requires a start date.
- hasEndDate [true|false]. If true the policy to create requires an end date.
- startDateLimit (String). An exact date, or a date expression to apply to the current time, to set an upper limit on the start date to use.
- endDateLimit (String). An exact date, or a date expression to apply to the current time, to set an upper limit on the end date to use.

The expression language used in startDateLimit and endDateLimit is the Date Math language used by Solr.

This new configuration allows us to simplify the presentation of pre-set options to the user such as access through the university network, embargoed, etc. and to offer new opportunities such as lease and temporal access.

JIRA Issues & PRs related with these changes

https://github.com/DSpace/DSpace/pull/1852 (merged)

https://github.com/DSpace/Rest7Contract/pull/11 (merged)

https://github.com/DSpace/DSpace/pull/1929 (merged)

https://github.com/DSpace/Rest7Contract/pull/16 (merged)

Unable to locate Jira server for this macro. It may be due to Application Link configuration.

Unable to locate Jira server for this macro. It may be due to Application Link configuration.

Unable to locate Jira server for this macro. It may be due to Application Link configuration.

Unable to locate Jira server for this macro. It may be due to Application Link configuration.

Unable to locate Jira server for this macro. It may be due to Application Link configuration.