Configurable Workflow

Introduction

Configurable Workflows are an optional feature that may be enabled for use only within DSpace XMLUI.

The primary focus of the workflow framework is to create a more flexible solution for the administrator to configure, and even to allow an application developer to implement custom steps, which may be configured in the workflow for the collection through a simple configuration file. The concept behind this approach was modeled on the configurable submission system already present in DSpace.

For more information, see the Configurable Workflow Introductory Video

Instructions for Enabling Configurable Reviewer Workflow in XMLUI

Please note that enabling the Configurable Reviewer Workflow makes changes to the structure of your database that are currently irreversible in any graceful manner, so please **backup your database** in advance to allow you to restore to that point should you wish to do so. It should also be noted that only the XMLUI has been changed to cope with the database changes. The JSPUI will no longer work if the Configurable Reviewer Workflow is enabled.

[dspace]/config/xmlui.xconf

The submission aspect has been split up into muliple aspects: one submission aspect for the submission process, one workflow aspect containing the code for the original workflow and one xmlworkflow aspect containing the code for the new XML configurable workflow framework. In order to enable one of the two aspects, either the workflow or xmlworkflow aspect should be enabled in the `[dspace]/config/xmlui.xconf` configuration file. This means that the xmlui.xconf configuration for the original workflow is the following:

```
<aspect name="Submission and Workflow" path="resource://aspects/Submission/" />
<aspect name="Original Workflow" path="resource://aspects/Workflow/" />
```

And the xmlui.xconf configuration for the new XML configurable workflow is the following:

```
<aspect name="Submission and Workflow" path="resource://aspects/Submission/" />
<aspect name="XMLWorkflow" path="resource://aspects/XMLWorkflow/" />
```

[dspace]/config/spring/api/core-services.xml

You also have to configure DSpace to load the right services. This is done by commenting the basic workflow services and uncommenting the xmlworkflow services in `[dspace]/config/spring/api/core-services.xml`. After enabling the configurable workflow the mentioned file should contain the following:

```
    <!--Basic workflow services, comment or remove when switching to the configurable workflow -->
    <!--
    <bean class="org.dspace.workflowbasic.TaskListItemServiceImpl"/>
    <bean class="org.dspace.workflowbasic.BasicWorkflowItemServiceImpl"/>
    <bean class="org.dspace.workflowbasic.BasicWorkflowServiceImpl"/>
    -->

    <!--Configurable workflow services, uncomment the xml workflow beans below to enable the configurable
workflow-->
    <bean class="org.dspace.xmlworkflow.storedcomponents.ClaimedTaskServiceImpl"/>
    <bean class="org.dspace.xmlworkflow.storedcomponents.CollectionRoleServiceImpl"/>
    <bean class="org.dspace.xmlworkflow.storedcomponents.InProgressUserServiceImpl"/>
    <bean class="org.dspace.xmlworkflow.storedcomponents.PoolTaskServiceImpl"/>
    <bean class="org.dspace.xmlworkflow.storedcomponents.WorkflowItemRoleServiceImpl"/>
    <bean class="org.dspace.xmlworkflow.storedcomponents.XmlWorkflowItemServiceImpl"/>
    <bean class="org.dspace.xmlworkflow.XmlWorkflowServiceImpl"/>
    <bean class="org.dspace.xmlworkflow.WorkflowRequirementsServiceImpl"/>
    <bean class="org.dspace.xmlworkflow.XmlWorkflowFactoryImpl"/>
```

Please be careful while editing `[dspace]/config/spring/api/core-services.xml` as it controls which parts of DSpace are loaded. Accidentally uncommenting the wrong parts my result in your DSpace instance not loading properly anymore.

[dspace]/config/spring/api/core-factory-services.xml

Same as for the core-services.xml from above, comment out the basic workflow factory & enable the xmlworkflow factory in `[dspace]/config/spring/api/core-factory-services.xml`. After enabling the configurable workflow the mentioned file should contain the following:

```
<!--Basic workflow services, comment or remove when switching to the configurable workflow -->
<!--<bean id="workflowServiceFactory" class="org.dspace.workflowbasic.factory.BasicWorkflowServiceFactoryImpl"
/>-->
<!--Configurable workflow services, uncomment to enable-->
<bean id="workflowServiceFactory" class="org.dspace.xmlworkflow.factory.XmlWorkflowServiceFactoryImpl"/>
```

[dspace]/config/hibernate.cfg.xml

With the xmlworkflow come some separate tables which hibernate needs to be aware of, and the tables for the basic workflow can be disregarded. So edit the `[dspace]/config/hibernate.cfg.xml` file and comment out the basic workflow classes & enable the xmlworkflow classes. The result is displayed below:

```
<!--<mapping class="org.dspace.workflowbasic.BasicWorkflowItem"/>-->
<!--<mapping class="org.dspace.workflowbasic.TaskListItem"/>-->
<mapping class="org.dspace.xmlworkflow.storedcomponents.ClaimedTask"/>
<mapping class="org.dspace.xmlworkflow.storedcomponents.CollectionRole"/>
<mapping class="org.dspace.xmlworkflow.storedcomponents.InProgressUser"/>
<mapping class="org.dspace.xmlworkflow.storedcomponents.PoolTask"/>
<mapping class="org.dspace.xmlworkflow.storedcomponents.WorkflowItemRole"/>
<mapping class="org.dspace.xmlworkflow.storedcomponents.XmlWorkflowItem"/>
```

Workflow Data Migration

You will also need to follow the Data Migration Procedure below.

Data Migration

Please note that enabling the Configurable Reviewer Workflow makes changes to the structure of your database that are currently irreversible in any graceful manner, so please **backup your database** in advance to allow you to restore to that point should you wish to do so. It should also be noted that only the XMLUI has been changed to cope with the database changes. The JSPUI will no longer work if the Configurable Reviewer Workflow is enabled.

Workflowitem conversion/migration scripts

Depending on the workflow that is used by a DSpace installation, different scripts can be used when migrating to the new workflow.

Automatic migration

Automatic migration can be used when the out of the box original workflow framework is used by your DSpace installation. This means that your DSpace installation uses the workflow steps and roles that are available out of the box. The automated migration will migrate the policies, roles, tasks and workflowitems from the original workflow to the new workflow framework.

Manually kick off this migration by simply running:

```
[dspace]/bin/dspace database migrate ignored
```

The "ignored" parameter will tell DSpace to run any previously-ignored migrations on your database. After enabling Configurable Workflow in your Spring configs (see above), the new automatic migrations will be made available to the "database migrate" command. As these new migrations were not previously run by "database migrate", they will be "ignored" until you trigger them by manually running the above command.

For more information on the "database migrate" command, please see Database Utilities.

Java based migration

In case your DSpace installation uses a customized version of the workflow, the migration script might not work properly and a different approach is recommended. Therefore, an additional Java based script has been created that restarts the workflow for all the workflowitems that exist in the original workflow framework. The script will take all the existing workflowitems and place them in the first step of the XML configurable workflow framework thereby taking into account the XML configuration that exists at that time for the collection to which the item has been submitted. This script can also be used to restart the workflow for workflowitems in the original workflow but not to restart the workflow for items in the XML configurable workflow.

To execute the script, run the following CLI command:

```
[dspace]/bin/dspace dsrun org.dspace.xmlworkflow.migration.RestartWorkflow -e admin@myrespository.org
```

The following arguments can be specified when running the script:

- -e: specifies the username of an administrator user
- -n: if sending submissions through the workflow, send notification emails
- -p: the provenance description to be added to the item
- -h: help

Configuration

Main workflow configuration

The workflow main configuration can be found in the workflow.xml file, located in **`[dspace]/config/workflow.xml`** . An example of this workflow configuration file can be found bellow.

```
<wf-config>
    <workflow-map>
        <!-- collection to workflow mapping -->
        <name-map collection="default" workflow="{workflow.id}"/>
        <name-map collection="123456789/0" workflow="{workflow.id2}"/>
    </workflow-map>

    <workflow start="{start.step.id}" id="{workflow.id}">
        <roles>
         <!-- Roles used in the workflow -->
        </roles>

            <!-- Steps come here-->
            <step id="ExampleStep1" nextStep="ExampleStep2" userSelectionMethod="{UserSelectionActionId}">
            <!-- Step1 config-->
            </step>
        <step id="ExampleStep2" userSelectionMethod="{UserSelectionActionId}">

        </step>
    </workflow>
    <workflow start="{start.step.id2}" id="{workflow.id}">
     <!-- Another workflow configuration-->
    </workflow>
</wf-config>
```

workflow-map

The workflow map contains a mapping between collections in DSpace and a workflow configuration. Similar to the configuration of the submission process, the mapping can be done based on the handle of the collection. The mapping with "default" as the value for the collection mapping, will be used for the collections not occurring in other mapping tags. Each mapping is defined by a "name-map" tag with two attributes:

- collection: can either be a collection handle or "default"
- workflow: the value of this attribute points to one of the workflow configurations defined by the "workflow" tags

workflow

The workflow element is a repeatable XML element and the configuration between two "workflow" tags represents one workflow process. It requires the following 2 attributes:

- id: a unique identifier used for the identification of the workflow and used in the workflow to collection mapping
- start: the identifier of the first step of the workflow, this will be the entry point of this workflow-process. When a new item has been committed to a collection that uses this workflow, the step configured in the "start" attribute will he the first step the item will go through.

roles

Each workflow process has a number of roles defined between the "roles" tags. A role represents one or more DSpace EPersons or Groups and can be used to assign them to one or more steps in the workflow process. One role is represented by one "role" tag and has the following attributes:

- id: a unique identifier (in one workflow process) for the role
- description: optional attribute to describe the role
- scope: optional attribute that is used to find our group and must have one of the following values:
    - collection: The collection value specifies that the group will be configured at the level of the collection. This type of groups is the same as the type that existed in the original workflow system. In case no value is specified for the scope attribute, the workflow framework assumes the role is a collection role.
    - repository: The repository scope uses groups that are defined at repository level in DSpace. The name attribute should exactly match the name of a group in DSpace.
    - item: The item scope assumes that a different action in the workflow will assign a number of EPersons or Groups to a specific workflow-item in order to perform a step. These assignees can be different for each workflow item.
- name: The name specified in the name attribute of a role will be used to lookup an eperson group in DSpace. The lookup will depend on the scope specified in the "scope" attribute:
    - collection: The workflow framework will look for a group containing the name specified in the name attribute and the ID of the collection for which this role is used.
    - repository: The workflow framework will look for a group with the same name as the name specified in the name attribute
    - item: in case the item scope is selected, the name of the role attribute is not required
- internal: optional attribute which isn't really used at the moment, false by default

```
<roles>
    <role id="{unique.role.id}" description="{role.description}" scope="{role.scope}" name="{role.name}"
internal="true/false"/>
</roles>
```

step

The step element represents one step in the workflow process. A step represents a number of actions that must be executed by one specified role. In case no role attribute is specified, the workflow framework assumes that the DSpace system is responsible for the execution of the step and that no user interface will be available for each of the actions in this step. The step element has the following attributes in order to further configure it:

- id: The id attribute specifies a unique identifier for the step, this id will be used when configuring other steps in order to point to this step. This identifier can also be used when configuring the start step of the workflow item.
- nextStep: This attribute specifies the step that will follow once this step has been completed under normal circumstances. If this attribute is not set, the workflow framework will assume that this step is an endpoint of the workflow process and will archive the item in DSpace once the step has been completed.
- userSelectionMethod: This attribute defines the UserSelectionAction that will be used to determine how to attache users to this step for a workflow-item. The value of this attribute must refer to the identifier of an action bean in the workflow-actions.xml. Examples of the user attachment to a step are the currently used system of a task pool or as an alternative directly assigning a user to a task.
- role: optional attribute that must point to the id attribute of a role element specified for the workflow. This role will be used to define the epersons and groups used by the userSelectionMethod.
- RequiredUsers

```
<step id="{step.id}" nextStep="{next.step.id}" userSelectionMethod="{user.selection.bean.id}" role="{role.id}" >
<!-- optional alternate outcomes, depending on the outcome of the actions you can alter the next step here -->
<alternativeOutcome>
     <step status="{integer}">{alternate.step.id}</step>
</alternativeOutcome>
<action id="{action.bean.id}"/>
<action id="{action.bean.id.1}"/>
</step>
```

Each step contains a number of actions that the workflow item will go through. In case the action has a user interface, the users responsible for the exectution of this step will have to execute these actions before the workflow item can proceed to the next action or the end of the step.

There is also an optional subsection that can be defined for a step part called "alternativeOutcome". This can be used to define outcomes for the step that differ from the one specified in the nextStep attribute. Each action returns an integer depending on the result of the action. The default value is "0" and will make the workflow item proceed to the next action or to the end of the step.
In case an action returns a different outcome than the default "0", the alternative outcomes will be used to lookup the next step. The alternativeOutcome element contains a number of steps, each having a status attribute. This status attribute defines the return value of an action. The value of the element will be used to lookup the next step the workflow item will go through in case an action returns that specified status.

Workflow actions configuration

API configuration

The workflow actions configuration is located in the `[dspace]/config/spring/api/` directory and is named "workflow-actions.xml". This configuration file describes the different Action Java classes that are used by the workflow framework. Because the workflow framework uses Spring framework for loading these action classes, this configuration file contains Spring configuration.

This file contains the beans for the actions and user selection methods referred to in the workflow.xml. In order for the workflow framework to work properly, each of the required actions must be part of this configuration.

```
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:util="http://www.springframework.org/schema/util"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans
/spring-beans-2.0.xsd
                        http://www.springframework.org/schema/util http://www.springframework.org/schema/util
/spring-util-2.0.xsd">

    <!-- At the top are our bean class identifiers --->
    <bean id="{action.api.id}" class="{class.path}" scope="prototype"/>
    <bean id="{action.api.id.2}" class="{class.path}" scope="prototype"/>

 <!-- Below the class identifiers come the declarations for out actions/userSelectionMethods -->

 <!-- Use class workflowActionConfig for an action -->
 <bean id="{action.id}" class="oorg.dspace.xmlworkflow.state.actions.WorkflowActionConfig" scope="prototype">
     <constructor-arg type="java.lang.String" value="{action.id}"/>

     <property name="processingAction" ref="{action.api.id}"/>
     <property name="requiresUI" value="{true/false}"/>
 </bean>

 <!-- Use class UserSelectionActionConfig for a user selection method -->
 <!--User selection actions-->
 <bean id="{action.api.id.2}" class="org.dspace.xmlworkflow.state.actions.UserSelectionActionConfig" scope="
prototype">
     <constructor-arg type="java.lang.String" value="{action.api.id.2}"/>

     <property name="processingAction" ref="{user.selection.bean.id}"/>
     <property name="requiresUI" value="{true/false}"/>
 </bean>
</beans>
```

Two types of actions are configured in this Spring configuration file:

- User selection action: This type of action is always the first action of a step and is responsible for the user selection process of that step. In case a step has no role attached, no user will be selected and the NoUserSelectionAction is used.
- Processing action: This type of action is used for the actual processing of a step. Processing actions contain the logic required to execute the required operations in each step. Multiple processing actions can be defined in one step. These user and the workflow item will go through these actions in the order they are specified in the workflow configuration unless an alternative outcome is returned by one of them.

User Selection Action

Each user selection action that is used in the workflow config refers to a bean definition in this workflow-actions.xml configuration. In order to create a new user selection action bean, the following XML code is used:

```
<bean id="{action.api.id.2}" class="org.dspace.xmlworkflow.state.actions.UserSelectionActionConfig" scope="
prototype">
    <constructor-arg type="java.lang.String" value="{action.api.id.2}"/>

    <property name="processingAction" ref="{user.selection.bean.id}"/>
    <property name="requiresUI" value="{true/false}"/>
</bean>
```

This bean defines a new UserSelectionActionConfig and the following child tags:

- constructor-arg: This is a constructor argument containing the ID the task. This is the same as the id attribute of the bean and is used by the workflow config to refer to this action.
- property processingAction: This tag refers the the ID of the API bean, responsible for the implementation of the API side of this action. This bean should also be configured in this XML.
- property requiresUI: In case this property is true, the workflow framework will expect a user interface for the action. Otherwise the framework will automatically execute the action and proceed to the next one.

Processing Action

Processing actions are configured similar to the user selection actions. The only difference is that these processing action beans are implementations of the WorkflowActionConfig class instead of the UserSelectionActionConfig class.

User Interface configuration

The configuration file for the workflow user interface actions is located in the `[dspace]/config/spring/xmlui/` and is named "workflow-actions-xmlui.xml". Each bean defined here has an id which is the action identifier and the class is a classpath which links to the xmlui class responsible for generating the User Interface side of the workflow action. Each of the class defined here must extend the `org.dspace.app.xmlui.aspect.submission.workflow.AbstractXMLUIAction` class, this class contains some basic settings for an action and has a method called `addWorkflowItemInformation()` which will render the given item with a show full link so you don't have to write the same code in each of your actions if you want to display the item. The id attribute used for the beans in the configuration must correspond to the id used in the workflow configuration. In case an action requires a User Interface class, the workflow framework will look for a UI class in this configuration file.

```
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:util="http://www.springframework.org/schema/util"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans
/spring-beans-2.0.xsd
                        http://www.springframework.org/schema/util http://www.springframework.org/schema/util
/spring-util-2.0.xsd">

    <bean id="{action.id}" class="{classpath}" scope="prototype"/>
    <bean id="{action.id.2}" class="{classpath}" scope="prototype"/>
</beans>
```

Authorizations

Currently, the authorizations are always granted and revoked based on the tasks that are available for certain users and groups. The types of authorization policies that is granted for each of these is always the same:

- READ
- WRITE
- ADD
- DELETE

Database

The workflow uses a separate metadata schema named `workflow`. The fields this schema contains can be found in the `[dspace]/config/registries` directory and in the file `workflow-types.xml`. This schema is only used when using the score reviewing system at the moment, but one could always use this schema if metadata is required for custom workflow steps.

The following tables have been added to the DSpace database. All tables are prefixed with 'cwf_' to avoid any confusion with the existing workflow related database tables:

cwf_workflowitem

The cwf_workflowitem table contains the different workflowitems in the workflow. This table has the following columns:

- workflowitem_id: The identifier of the workflowitem and primary key of this table
- item_id: The identifier of the DSpace item to which this workflowitem refers.
- collection_id: The collection to which this workflowitem is submitted.
- multiple_titles: Specifies whether the submission has multiple titles (important for submission steps)
- published_before: Specifies whether the submission has been published before (important for submission steps)
- multiple_files: Specifies whether the submission has multiple files attached (important for submission steps)

cwf_collectionrole

The cwf_collectionrole table represents a workflow role for one collection. This type of role is the same as the roles that existed in the original workflow meaning that for each collection a separate group is defined to described the role. The cwf_collectionrole table has the following columns:

- collectionrol_id: The identifier of the collectionrole and the primaty key of this table
- role_id: The identifier/name used by the workflow configuration to refer to the collectionrole
- collection_id: The collection identifier for which this collectionrole has been defined
- group_id: The group identifier of the group that defines the collection role

cwf_workflowitemrole

The cwf_workflowitemrole table represents roles that are defined at the level of an item. These roles are temporary roles and only exist during the execution of the workflow for that specific item. Once the item is archived, the workflowitemrole is deleted. Multiple rows can exist for one workflowitem with e.g. one row containing a group and a few containing epersons. All these rows together make up the workflowitemrole The cwf_workflowitemrole table has the following columns:

- workflowitemrole_id: The identifier of the workflowitemrole and the primaty key of this table

- role_id: The identifier/name used by the workflow configuration to refer to the workflowitemrole
- workflowitem_id: The cwf_workflowitem identifier for which this workflowitemrole has been defined
- group_id: The group identifier of the group that defines the workflowitemrole role
- eperson_id: The eperson identifier of the eperson that defines the workflowitemrole role

cwf_pooltask

The cwf_pooltask table represents the different task pools that exist for a workflowitem. These task pools can be available at the beginning of a step and contain all the users that are allowed to claim a task in this step. Multiple rows can exist for one task pool containing multiple groups and epersons. The cwf_pooltask table has the following columns:

- pooltask_id: The identifier of the pooltask and the primaty key of this table
- workflowitem_id: The identifier of the workflowitem for which this task pool exists
- workflow_id: The identifier of the workflow configuration used for this workflowitem
- step_id: The identifier of the step for which this task pool was created
- action_id: The identifier of the action that needs to be displayed/executed when the user selects the task from the task pool
- eperson_id: The identifier of an eperson that is part of the task pool
- group_id: The identifier of a group that is part of the task pool

cwf_claimtask

The cwf_claimtask table represents a task that has been claimed by a user. Claimed tasks can be assigned to users or can be the result of a claim from the task pool. Because a step can contain multiple actions, the claimed task defines the action at which the user has arrived in a particular step. This makes it possible to stop working halfway the step and continue later. The cwf_claimtask table contains the following columns:

- claimtask_id: The identifier of the claimtask and the primary key of this table
- workflowitem_id: The identifier of the workflowitem for which this task exists
- workflow_id: The id of the workflow configuration that was used for this workflowitem
- step_id: The step that is currenlty processing the workflowitem
- action_id: The action that should be executed by the owner of this claimtask
- owner_id: References the eperson that is responsible for the execution of this task

cwf_in_progress_user

The cwf_in_progess_user table keeps track of the different users that are performing a certain step. This table is used because some steps might require multiple users to perform the step before the workflowitem can proceed. The cwf_in_progress_user table contains the following columns:

- in_progress_user_id: The identifier of the in progress user and the primary key of this table
- workflowitem_id: The identifier of the workflowitem for which the user is performing or has performed the step.
- user_id: The identifier of the eperson that is performing or has performe the task
- finished: Keeps track of the fact that the user has finished the step or is still in progress of the execution

Additional workflow steps/actions and features

Optional workflow steps: Select single reviewer workflow

This workflow makes it possible to assign a single user to review an item. This workflow configuration skips the task pool option meaning that the assigned reviewer no longer needs to claim the task. The configuration consists of the following 2 steps.

- AssignStep: During the assignstep, a user has the ability to select a responsible user to review the workflowitem. This means that for each workflowitem, a different user can be selected. Because a user is assigned, the task pool is no longer required.
- ReviewStep: The start of the reviewstep is different than the typical task pool. Instead of having a task pool, the user will be automatically assigned to the task. However, the user still has the option to reject the task (in case he or she is not responsible for the assigned task) or review the item. In case the user rejects the task, the workflowitem will be sent to the another step in the workflow as an alternative to the default outcome.

Optional workflow steps: Score review workflow

The score review system allows reviewers to give the reviewed item a rating. Depending on the results of the rating, the item will be approved to go to the next workflow step or will be sent to an alternative step. The scrore review workflow consists of the following 2 steps.

- ScoreReviewStep: The group of responsible users for the score reviewing will be able to claim the task from the taskpool. Dependingn on the configuration, a different number of users can be required to execute the task. This means that the task will be available in the task pool until the required number of users has at least claimed the task. Once everyone of them has finished the task, the next (automatic) processing step is activated.
- EvaluationStep: During the evaluationstep, no user interface is required. The workflow system will automatically execute the step that evaluates the different scores. In case the average score is more than a configurable percentage, the item is approved, otherwise it is rejected. (The minimum average score is set by adjusting the `minimumAcceptanceScore` property passed to `evaluationactionAPI` in `config/spring/api/workflow-actions.xml`.)

Workflow overview features

A new features has been added to the XML based workflow that resembles the features available in the JSPUI of DSpace that allows administrators to abort workflowitems. The feature added to the XMLUI allows administrators to look at the status of the different workflowitems and look for workflowitems based on the collection to which they have been submitted. Besides that, the administrator has the ability to permanently delete the workflowitem or send the item back to the submitter.

Known Issues

Curation System

The DSpace 1.7 version of the curation system integration into the original DSpace workflow only exists in the WorkflowManager.advance() method. Before advancing to the next workflow step or archiving the Item, a check is performed to see whether any curation tasks need to be executed/scheduled. The problem is that this check is based on the hardcoded workflow steps that exist in the original workflow. These hardcoded checks are done in the WorkflowCurator and will need to be changed.

Existing issues

- What happens with collection roles after config changes
- What with workflowitems after config changes
- What with undefined outcomes
- Config checker
- Configurable authorizations?