

Grouper UI - Templates

Wiki Home	Grouper Release Announcements	Grouper Guides	Grouper Deployment Guide	Community Contributions	Internal Developer Resources
---------------------------	---	--------------------------------	--	---	--

Grouper UI templates are available in Grouper 2.4+.

The UI templates allow folder admins to run a function. The template creates folders and groups and assigns privileges based on inputs. This will allow Grouper users to accomplish multiple tasks at once and save time and be more consistent. The original template implementation was Java-based. In 2.5 + templates can be implemented with configuration and GSH scripts (Java/groovy).

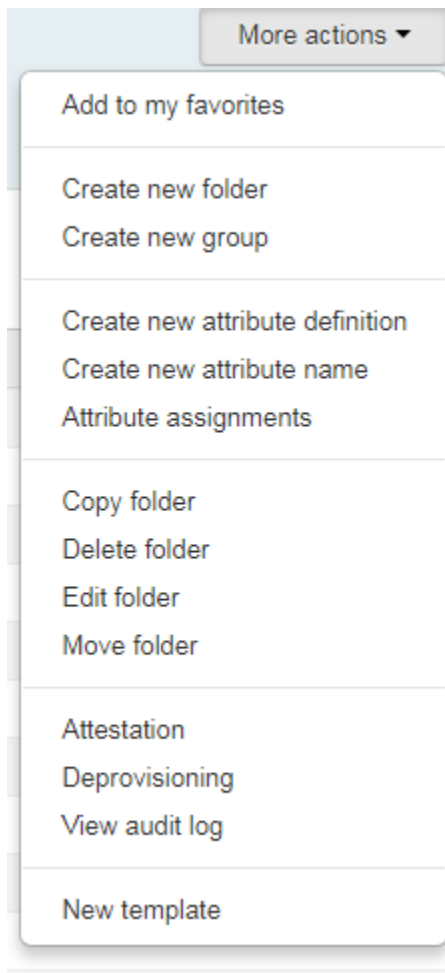


Using the template wizard provides a convenient approach to implementing the recommendations for folder and group structure from the [Grouper Deployment Guide](#).

- [Grouper UI - Template wizard - Policy group](#)
- [Using the Template Wizard](#)
- [Application built-in template](#)
- [Grouper Deployment Guide structure built-in template](#)
- [Creating your own template \(GSH\)](#)
- [Creating your own template \(legacy\)](#)
- [Run GSH template from GSH](#)
- [Run template from GSH \(legacy\)](#)

Using the Template Wizard

On a folder screen, under "more actions" will be an option for "Run template"



The wizard will prompt for "Template type", which by default has "Application" and "Grouper Deployment Guide structure."

[Home](#) > [Root](#)

 **Root**

[Edit folder](#)

[More actions](#) ▼

[More](#) ▼

[Folder contents](#)

[Privileges](#)

[More](#) ▼

Template type

Application

Grouper Deployment Guide structure

Create in this folder, do not create a subfolder

Key

*

Alphanumeric key for template being run. Might be used for object names.

Friendly Name

☐ Edit friendly name

This will typically be the same as the key though could be more friendly. Might be used for object display names.

Description

Describe the purpose that the template is being run. Might be used on object descriptions.

[Next](#)

[Cancel](#)

Application built-in template

The "application" option allows you to give a friendly name to an application and prompts you to set up the recommended folders for an application to be integrated with Grouper.

Template type Application
Pick the template type you want to run

Create in this folder, do not create a subfolder ☐

Key issueTracker *
Alphanumeric key for template being run. Might be used for object names.

Friendly Name issueTracker
☐ Edit friendly name
This will typically be the same as the key though could be more friendly. Might be used for object display names.

Description The institution's issue tracker
Describe the purpose that the template is being run. Might be used on object descriptions.

Next Cancel

☒ Create folder: "app:issueTracker"?

☒ Create folder: "app:issueTracker:service"?

☒ Create folder: "app:issueTracker:service:policy"?

☒ Create folder: "app:issueTracker:service:ref"?

☒ Create folder: "app:issueTracker:service:attribute"?

☒ Create folder: "app:issueTracker:security"?

☒ Create group "app:issueTracker:security:issueTrackerAdmins"?

☒ Assign "app:issueTracker:security:issueTrackerAdmins" to have inherited ADMIN privilege on Groups on the "app:issueTracker" folder?

☒ Assign "app:issueTracker:security:issueTrackerAdmins" to have inherited ADMIN privilege on Folders on the "app:issueTracker" folder?

☒ Assign "app:issueTracker:security:issueTrackerAdmins" to have inherited ADMIN privilege on Attributes on the "app:issueTracker" folder?

☒ Create group "app:issueTracker:security:issueTrackerReaders"?

☒ Assign "app:issueTracker:security:issueTrackerReaders" to have inherited READ privilege on Groups on the "app:issueTracker" folder?

☒ Create group "app:issueTracker:security:issueTrackerUpdaters"?

☒ Assign "app:issueTracker:security:issueTrackerUpdaters" to have inherited UPDATE privilege on Groups on the "app:issueTracker:service" folder?

☒ Assign "app:issueTracker:security:issueTrackerUpdaters" to be a member of "app:issueTracker:security:issueTrackerReaders"?

Grouper Deployment Guide structure built-in template

This is a convenient way to implement the recommendations for folder structure from the [Grouper Deployment Guide](#).

Home > Root

Root

More ▾

Folder contentsPrivilegesMore ▾

Edit folder

More actions ▾

Template type

Grouper Deployment Guide structure ▾

Pick the template type you want to run

Create in this folder, do not create a subfolder

☒

Next

Cancel

- ☒ Create folder: "basis"?
 - ☒ Create folder: "ref"?
 - ☒ Create folder: "bundle"?
 - ☒ Create folder: "app"?
 - ☒ Create folder: "org"?
 - ☒ Create folder: "test"?
 - ☒ Create folder: "etc"?
 - ☒ Create folder: "etc:security"?
 - ☒ Create group "etc:security:Admins"?
 - ☒ Assign "etc:security:Admins" to have inherited ADMIN privilege on Groups on the "" folder?
 - ☒ Assign "etc:security:Admins" to have inherited ADMIN privilege on Folders on the "" folder?
 - ☒ Assign "etc:security:Admins" to have inherited ADMIN privilege on Attributes on the "" folder?
 - ☒ Create group "etc:security:Readers"?
 - ☒ Assign "etc:security:Readers" to have inherited READ privilege on Groups on the "" folder?
 - ☒ Create group "etc:security:Updaters"?
 - ☒ Assign "etc:security:Updaters" to have inherited UPDATE privilege on Groups on the "basis" folder?
 - ☒ Assign "etc:security:Updaters" to have inherited UPDATE privilege on Groups on the "reference" folder?
 - ☒ Assign "etc:security:Updaters" to have inherited UPDATE privilege on Groups on the "bundle" folder?
 - ☒ Assign "etc:security:Updaters" to have inherited UPDATE privilege on Groups on the "application" folder?
 - ☒ Assign "etc:security:Updaters" to have inherited UPDATE privilege on Groups on the "organization" folder?
 - ☒ Assign "etc:security:Updaters" to have inherited UPDATE privilege on Groups on the "test" folder?
 - ☒ Assign "etc:security:Updaters" to be a member of "etc:security:Readers"?
-

Creating your own template (GSH)

See the [GSH templates for examples](#)

Creating your own template (legacy)

New templates should be configured in grouper-ui.properties, and should extend a common base class. GrouperTemplateLogicBase

- Base class
- In Java will configure options for new service

See the built-ins for examples

```
#####  
##### New service template  
#####  
  
## use capturing group from regex as the key in the dropdown. eg: service for the below string  
grouper.template.service.logicClass=edu.internet2.middleware.grouper.grouperUi.beans.ui.  
GrouperNewServiceTemplateLogic  
  
grouper.template.tierStructure.logicClass=edu.internet2.middleware.grouper.grouperUi.beans.ui.  
GrouperTierStructureLogic
```

The wizard will prompt them for a "key" (alphanumeric), and friendly name (optional). Also the service description.

The base class should take as an input a stem, a system name extension, and optionally a friendly name extension (default to system name extension)

There should be a ui text key that describes the service (e.g. built in is "New service"). Drop down will pick which type of service

It should return a list of things it will do, as javabeans

ServiceAction

- reference back to service (which has reference to stem name, and system and display extensions)
- indentLevel (if should be indented on screen to make easier to read, e.g. under a new stem)
- type: stem, group, membership, privilege, inheritedPrivilege, attributeDef, attributeName, attributeAssignment
- arg0: e.g. for stem would be EL for the name, e.g. \${parentStemName}:\${serviceSystemName}etc. e.g. for privilege is the privilegeName
- arg1
- arg2
- externalized key for the label
- defaultChecked (true|false)
- checkSubmitted (if the checkbox was checked by user)

The UI should display the ServiceActions on the screen, with the correct indent level, and checkboxes

When submitted, if each checkbox is checked, it should do that action (if applicable). If not applicable, give a warning. The UI will pass the list of pojos, and "checkSubmitted" is set based on if the checkboxes were checked, to the base class, for a logic method, to implement the template. There should be a base class method to validate the input, pass the response JS object, and the validate method can write error messages to the screen and return true or false if valid.

Run GSH template from GSH

See the [GSH attribute sync example](#)

Run template from GSH (legacy)

Note: GSH must be run from the UI. This works in v4.1.5+

```

String templateType = "service";
String stemName = "test";
String templateKey = "app3";
String serviceDescription = "app3 app";
String serviceFriendlyName = "app3";
boolean createSubFolder = true;
GrouperSession grouperSession = GrouperSession.startRootSession();
Stem stem = StemFinder.findByName(grouperSession, stemName, true);
edu.internet2.middleware.grouper.grouperUi.beans.ui.GrouperRequestContainer.assignUseStaticRequestContainer
(true);
edu.internet2.middleware.grouper.grouperUi.beans.ui.GrouperRequestContainer grouperRequestContainer = edu.
internet2.middleware.grouper.grouperUi.beans.ui.GrouperRequestContainer.retrieveFromRequestOrCreate();
edu.internet2.middleware.grouper.grouperUi.beans.ui.GroupStemTemplateContainer groupStemTemplateContainer =
grouperRequestContainer.getGroupStemTemplateContainer();
edu.internet2.middleware.grouper.grouperUi.beans.ui.GrouperTemplateLogicBase templateLogic = edu.internet2.
middleware.grouper.grouperUi.serviceLogic.UiV2Template.getTemplateLogic(templateType,
groupStemTemplateContainer);
templateLogic.setStemId(stem.getUuid());
groupStemTemplateContainer.setCreateNoSubfolder(!createSubFolder);
groupStemTemplateContainer.setTemplateKey(templateKey);
groupStemTemplateContainer.setTemplateDescription(serviceDescription);
groupStemTemplateContainer.setTemplateFriendlyName(serviceFriendlyName);
List<edu.internet2.middleware.grouper.grouperUi.beans.ui.ServiceAction> allServiceActions = templateLogic.
getServiceActions();
// if (templateLogic.validate(allServiceActions)) { throw new RuntimeException("Not valid"); } // dont
include this line for some reason
for (edu.internet2.middleware.grouper.grouperUi.beans.ui.ServiceAction serviceAction: allServiceActions) {
serviceAction.getServiceActionType().createTemplateItem(serviceAction); }

```