**SailPoint**

**So, You want to contribute to SERI?**

**Turning your work into great reusable artefacts**

**Kevin James**

[So, you want to contribute to SERI? 2](#_Toc359239383)

[Working on your contribution 2](#_Toc359239384)

[Create a branch and switch to it 2](#_Toc359239385)

[Build your contribution 3](#_Toc359239386)

[How to name your artefacts 3](#_Toc359239387)

[Catalog Entry 3](#_Toc359239388)

[Artefacts 3](#_Toc359239389)

[How to import your artefacts 3](#_Toc359239390)

[Merging artefacts 4](#_Toc359239391)

[Do many, many tests 4](#_Toc359239392)

[Document your work 4](#_Toc359239393)

[Merge your branch into the trunk 4](#_Toc359239394)

[Make sure you have committed all your changes 4](#_Toc359239395)

[Update your branch with the latest updates from trunk 5](#_Toc359239396)

[Test again 5](#_Toc359239397)

[Switch to trunk 5](#_Toc359239398)

[Reintegrate your changes 5](#_Toc359239399)

[Beer Time! 5](#_Toc359239400)

[Creating a Merge artefact 5](#_Toc359239401)

[Resolving SVN conflicts 5](#_Toc359239402)

# So, you want to contribute to SERI?

Congratulations! You are joining an elite group of people – a group of people who take their hard work, prepare it for consumption and understanding by the larger group, document it and then share it! You should feel a warm glow inside, but don’t pat yourself on the back just yet… there is work to be done!

## Working on your contribution

The SERI project is stored in a Subversion repository. You can get the latest and greatest (also known as “the trunk”) from <https://svn.sailpoint.com/svn/sales/seri>. This is where your contribution will eventually end up, but to start with we’ll need to create a “branch”. A branch is a copy of the project that you can modify and share without modifying the trunk. This is useful if you want to, for example, transfer your work in progress between computers, or work collaboratively on a contribution without “breaking the trunk”. Breaking the trunk is when you check in a change which means that the project will no longer compile, or an existing contribution will no longer work. Ideally you will:

* Create a branch
* Switch to that branch
* Work on your contribution
* Do many, many tests
* Merge your branch into the trunk

### Create a branch and switch to it

Subversion is generally worked on from the command line. You will probably have already checked out a copy of the project with

$ svn co <https://svn.sailpoint.com/svn/sales/seri>

If not, you’ll need to do that now so we can create a copy. If you have already got a copy, it’s best to make sure you have the latest with

$ svn update

Now we can create a branch and switch to it. Branches live in seri-branches, so we need to create a copy in there

The format for the branch name is <initials>-<feature you’re working on>

So, let’s create the branch and switch our working copy to it:

$ svn copy https://svn.sailpoint.com/svn/sales/seri https://svn.sailpoint.com/svn/sales/seri-branches/kmj-docFeature

$ svn switch https://svn.sailpoint.com/svn/sales/seri-branches/kmj-docFeature

Our changes will now go to the branch rather than the trunk when we commit them.

### Build your contribution

Now the hard work begins! You will already know what you want to contribute. If you are going to build on top of a new install of SERI, once your environment is ready to go you can use the following command to set a point in time.

$ ant set-snapshot

This will make a record of “now” in your build.properties file. Later, when you have finished building your catalog entry you can use the following command to export only artifacts that have changed since this fixed point.

$ ant export-snapshot

You may even already have some artefacts ready to copy over. So that SERI remains usable and uncluttered, there are some naming guidelines you will need to follow.

#### How to name your artefacts

Firstly, everything is stored in the project under the directory config/catalog. We are using the paradigm of a catalog, where you can select individual items to tailor your environment. Each catalog entry is self-contained so that you know when you select a catalog entry, you will be getting all the artefacts required for that catalog entry.

##### Catalog Entry

You will be creating a directory for your catalog entry, the name must start with one of these:

|  |  |
| --- | --- |
| Start with this.. | When you are building.. |
| Connector | A new type of connector that will be available in the list in Define->Applications |
| Resource | A new Application definition (with rules etc. if required) for use in a demo environment |
| UseCase | A set of artefacts to show a specific use case, which could be used in the SERI Demo environment or during a POC |
| Utility | Anything that may be used to make a task easier, for example importing a file to generating objects in IdentityIQ. |

Now you have the stalk of your directory name, you will add a hypen and a camel-cased name that describes your contribution. For example

* UseCase-DeferredTerminate
* Utility-ChallengeAnswerImporter

##### Artefacts

The artefacts that you build must also follow a naming convention. This is ObjectType-UpperCamelCasedName.xml. The name is typically the internal name of the object – for example, a Rule called “BuildMap – Active Directory” would be defined in a file called Rule-BuildMap-ActiveDirectory.xml

#### How to import your artefacts

Once you have all your artefacts defined, you’ll need some way for a user to import them. Requiring them to know all the artefacts for a catalog entry seemed to be unfair, so we came up with the idea of having a single file in each catalog entry that needs to be imported. This is setup.xml.

Your setup.xml will need to look like this

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE sailpoint PUBLIC 'sailpoint.dtd' 'sailpoint.dtd'>

<sailpoint>

*Import actions go here..*

</sailpoint>

The import actions are relative to the current directory (your artefact). So, if your catalog entry relies on an artefact from another, you can refer to it with something link ../otherCatalogEntry/artefact.xml

For each of your artefacts, you will need an import entry. This will look like:

<ImportAction name='include' value='filename.xml'/>

##### Merging artefacts

Most of the time, your artefacts will be discrete, whole objects (for example, a Rule or a Workflow). However, sometimes you will want to update an existing object. A prime example of this is when you want to add an new Identity Mapping.

For this, you will need to create a merge artefact. See “Creating a merge artefact”

### Do many, many tests

Do some testing. Still need to write this, but basically rebuild SERI, deploy it, test your work. Modify and retest until it works

### Document your work

You will need to provide a readme.txt file with your catalog entry. It needs to look like this:

NameOfThis v x.y.z

Date of this revision/version

Contact: person.who.made.it@sailpoint.com

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Library dependencies \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If it’s a utility, any java libraries that it depends on

\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Execution \*

\*\*\*\*\*\*\*\*\*\*\*\*\*

If it’s a utility, how to run it. Command line? Utility rule run from debug/console?

\*\*\*\*\*\*\*\*\*\*\*\*

\* Background \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A short paragraph about why this use case was developed; what niche does it fill? What was the use case from the customer?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Potential Enhancements \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Anything you might want to add in the future (or someone else might want to add)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Limitations \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Anything that someone using it might need to be aware of, for example “doesn’t work with manual workitems” or “attribute XYZ is hardcoded”

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* To Configure \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

How to configure this. Probably most appropriate for use cases. Start with the import, then explain any configuration parameters in rules, workflows etc. Does the user have to run any tasks before it’s usable? Basically the steps to get it ready to demonstrate.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* To Demonstrate \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A simple set of steps to go through in order to demonstrate the use case

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Version History \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

21 Jun 2013

What I changed since the previous one

24 Apr 2013

Integration into SERI

08 Feb 2013

First release

### Merge your branch into the trunk

Now comes the fun part – merging your work back into the trunk.

#### Make sure you have committed all your changes

To make sure you have a “clean” project, “svn status” should return nothing:

$ svn status

$

If it doesn’t, then you need to do a commit, or remove some files. A ‘?’ denotes a file that is unknown (a new file that you have not added to version control), or a ‘M’ denotes a file that has been modified:

For example, if you get this:

kev@kj-desktop /workspace-juno/seri

$ svn status

M build.xml

? newFile

You would need to

kev@kj-desktop /workspace-juno/seri

$ svn add newFile

Then

kev@kj-desktop /workspace-juno/seri

$ svn commit

#### Update your branch with the latest updates from trunk

Once svn status returns nothing, you are up to date submitting your changes. Now you need to bring in anything that might have changed in the trunk:

$ svn merge ^/sales/seri (^ is a shortcut to the root of the SVN tree)

--- Merging r45114 through r45418 into '.':

A config\xml\Rule\Rule-Query-IdentityAttributeValues.xml

A config\catalog\Demo-Standard\Configuration-PrivilegedActiveDirectoryGroups.xml

D config\catalog\Demo-Standard\Configuration\_PrivilegedActiveDirectoryGroups.xml

U config\catalog\Resource-ActiveDirectory\Application-AD.xml

U scripts\build.dev.xml

--- Recording mergeinfo for merge of r45114 through r45418 into '.':

U .

Here you can see that some of the files have changed since we branched from the trunk. We have now got the latest trunk with all of our branch changes. If you have any conflicts, you will need to address them. See Appendix B

#### Test again

Now you will need to test again.

#### Reintegrate your changes

Once your testing succeeds, check everything back into subversion. This will include the updates from trunk and anything you fixed during your testing.

$ svn commit

Now we need to switch back to trunk. Use the command

$ svn switch ^/sales/seri

Then reintegrate the changes from your branch with

$ svn merge --reintegrate ^/sales/seri-branches/kmj-docFeature

Now you need to resolve any conflicts, and commit the changes

$ svn status

M .

A + config\ExtendedAttributes

M config\catalog\Resource-HR\ObjectConfig-Identity.xml

M scripts\build.config.xml

$ svn commit -m "Merged changed from branch kmj-docFeature"

Sending .

Adding config\ExtendedAttributes

Sending config\catalog\Resource-HR\ObjectConfig-Identity.xml

Sending scripts\build.config.xml

Transmitting file data ..

Committed revision 45420.

## Beer Time!

Well done! You’ve now contributed something that other people can use for years to come. Time for a beer!

Creating a Merge artefact

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE sailpoint PUBLIC "sailpoint.dtd" "sailpoint.dtd">

<sailpoint>

<ImportAction name='merge'>

Resolving SVN conflicts

See <http://svnbook.red-bean.com/en/1.6/svn.tour.treeconflicts.html>