



CA Release Automation
Shared Components
Best Practice & Coding Standards

Author : Walter Guerrero – walter.guerrero@ca.com
Version: 1.7
Filename: CA-RA-SharedComponents-Coding-Standards.docx
Date: 4/6/2015

Table of Contents

Introduction	4
What is a Shared Component	4
How to create shared component	5
Step 1 – Shared Component Planning	5
Step 2 – Actions to be used.....	6
Step 3 – Flows to be used	6
Step 4 – Software Artifacts to be used	7
Step 5 – Server Types Assignment	7
Step 6 – Using shared component s in applications	8
Coding Naming Conventions.....	10
Components Names.....	10
Flow Names.....	11
Actions Name.....	11
Parameters Names.....	11
Internal.....	12
Environment.....	12
Release	12
User Input	13
Collection Names	13
Artifacts.....	13
Exporting Shared Components	13
Documentation Conventions	14
Release Doc.....	14
Descriptions in Shared Components.....	14
Best Practices.....	16
Shared Components.....	16
Multiple Revisions.....	16
Mapped Artifact Types.....	17
Shared Component’s Actions.....	17
Shared Component’s Flows	17
Generic Flows.....	18
Number of actions in flows	18
Loops in Flows.....	19

Setting Booleans as result of flow.....	20
Copyright Notice	20

Introduction

This document will focus on the Best Practices, How To's and coding standards that will provide guidance to create reusable shared components. This new capability allows users the ability to create workflows much faster with pre-built, tested workflows that follow coding standards.

These coding standards will make the Release Automation flows created easier to understand, modify, and debug.

What is a Shared Component

Shared components provide users with the ability of creating deployment logic once and share the deployment logic across multiple Release Automation applications. Version control is an integral part of shared components, where you can have multiple revisions available to the Release Automation application. This is very critical, when you would like to maintain application a at the shared component revision b and application b at the shared component revision 4. Shared components help organizations establish best practices by defining standards regarding common deployments.

Using shared components ensures that all teams follow the same guidelines and workflows. For example, if you have multiple applications that use a Tomcat server, define the flows in a shared component and use it in all of these Release Automation applications.

How to create shared component

The creation of a shared component needs to follow a very methodical process; we have created a best practices list (shown below) to help in the proper “flow” to follow for a successful development of a shared component.

- Step 1 – Shared component planning
- Step 2 – Actions to be used
- Step 3 – Flows to be used
- Step 4 – Software Artifacts to be used
- Step 5 – Server type assignment
- Step 6 – Using shared components in applications

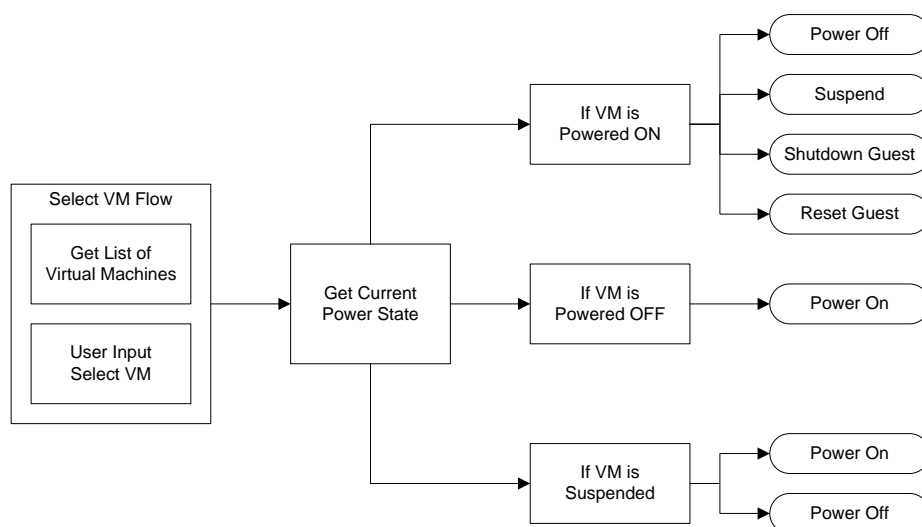
After the above steps have been completed (which will be expanded below) , you will package the shared component for distribution or make the shared component available to the necessary applications in the present Release Automation installation.

Step 1 – Shared Component Planning

Prior to the creation of the shared component, you need to plan out what exactly is the shared component going to be doing, as well as the number of action packs that you will be utilizing as part of this shared component.

Part of this planning will be the creation of a flow chart showing how the different actions and flows will be interacting for the creation of the given shared component.

The objective is once the shared component has been created and a “release” version is made available, *you will be able to build out processes based off these flows in an application within 10 minutes (based on complexity).*



The above flow chart shows an example of the types of steps that you will need to take for the powering on/off of a VMware VM image.

Step 2 – Actions to be used

Once you have planned out the purpose of the shared component and what exactly you need the shared component to accomplish. Follow the procedure below.

- Look at the generic actions that you will need, for example set parameters, find files, manipulate files, etc.
- Select the correct action pack(s) that will be needed in the shared component.
- Under the “Actions”, it is recommended that you define the following categories, these categories represents the capabilities associated for each section, for example for the “Getters”, you will add actions that get a value, i.e. “Get file status”. You need to also include a section in relation to the exclusive actions for this shared component, such as JBoss.
 - Getters
 - Setters
 - Checkers
 - Generic
 - <SharedComponent>
- Defined the parameters that you will be using, it is recommended that you setup the proper environment and release variables to be used as the main interaction via the deployment phase.
- Look at values that are constantly being used and create the parameters based on the parameter conventions listed below.
- In a shared component, you will need to utilize parameter values to interact with the actions and flows that make up the given shared component. This become more important, since you can only pass parameters and you can only update the parameters at the action level, you cannot update parameters or entries at the flow level.
- Once you have added the shared component to a given application, you can update the values associated with the given shared component parameters.
-

Step 3 – Flows to be used

Once you defined and named the actions that will become part of the shared components, it is recommended that you also create categories for the flows as well. This improves readability and maintainability of the flows.

- Create the following categories
 - Generic
 - Management
 - Configuration
- Based on what the shared components will be performing, you can create additional categories as needed.
- Create generic flows that perform commonly used tasks, for example a flow that will provide the JBoss home dir, instead of having to enter this value via an environment or release

parameter, this value is determined at runtime and provided to all the other flows as needed.

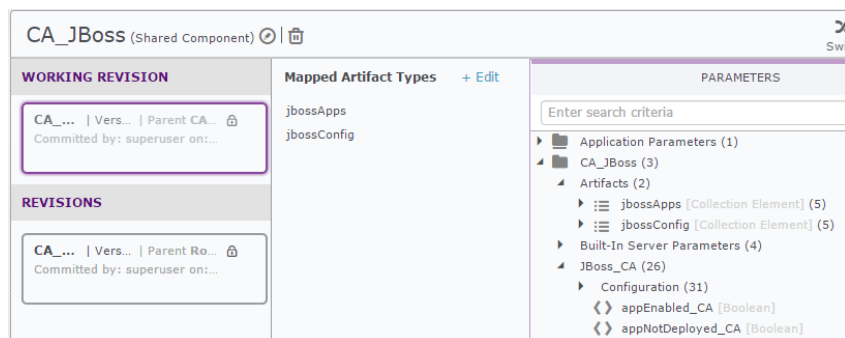
- For the processing of global artifacts, it is recommended that you create a sub-flow that will be using a loop based off the collection of the given artifact.

Step 4 – Software Artifacts to be used

Shared components¹ can also use artifacts. Below is a definition of global artifact.

A global artifact is an artifact that has been mapped to a shared component, where it can be ported to different applications in different Release Automation installations.

The importance of global artifacts is its portability and the processing of the artifacts defined within the global artifact.



As shown in the image above, you can see that the “mapped” artifact type has been assigned to the JBoss shared component. The recommendation here is to look at the type of artifacts that will be generic enough to all the flows that make up the shared component.

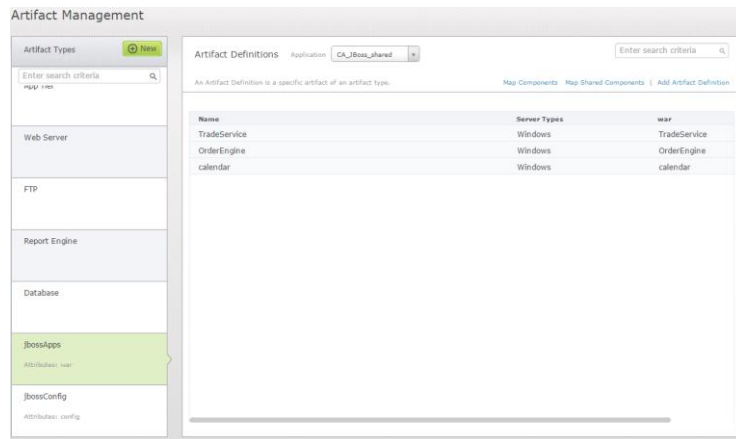
For example, “jbossApps” and “jbossConfig” are the defined global artifact types for this shared component.

If the global artifact does not exist, when you import the shared component into a Release Automation installation, the global artifact will be created for you. You still need to populate the definition types and release versions.

Step 5 – Server Types Assignment

Once you have added the necessary artifact definition to the global artifact that was installed as part of the shared component, and you have made the shared component part of an application, it is important that assigned the necessary server type to those artifact definitions, below is an example of what this would look like.

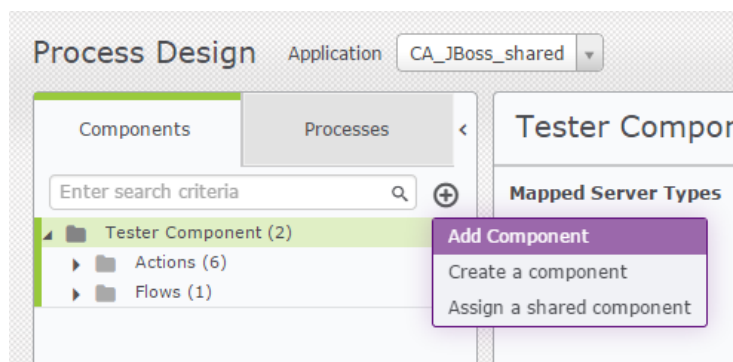
¹ Available starting with RA version 5.5.1.



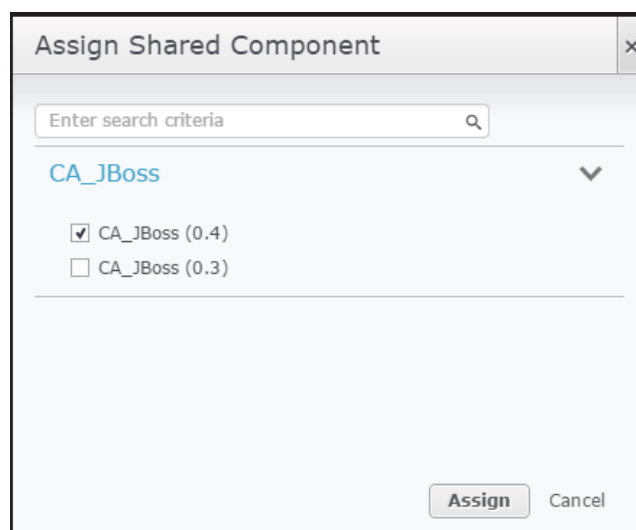
Step 6 – Using shared components in applications

Once the shared component has been created, you will have two methods that you can follow.

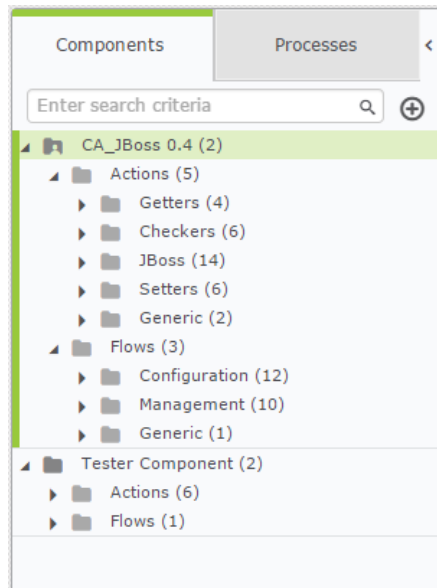
- Using the shared component in the same Release Automation installation, this is accomplished by assigning the shared component to the application as shown below.



In the design process, you will “assign a shared component” as shown in the image above.



The working revision should be the highest version listed, in the image above that will be revision 0.4.

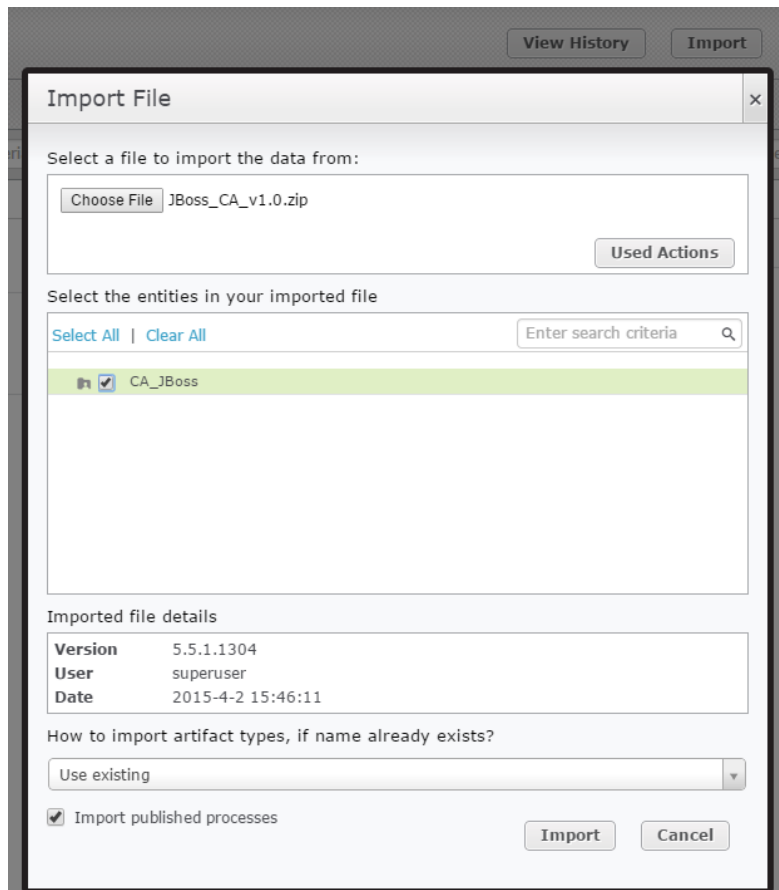


Finally, you are going to see the shared component that is now part of the application, you are now ready to either utilize the pre-built shared component's flows/actions or use them to create additional flows in the application's processes.

Now, if you need to distribute this shared component, you will have to export the shared component as described in the "Export Shared Component" section below.

- Once you have the created zipfile, you need to login to the new Release Automation installation.
- Go to the Designer→Import/Export link.
- Click the "Import" button and proceed to import the shared component zipfile.

See the image below to see what this would look like.



Coding Naming Conventions

It is important that **meaningful names** be used to name the actions, flows, parameters, and collections that will make up the shared components libraries.

We are going to use **lower camel-casing**, where the first letter of each embedded word is lower case, with the exception of the technology that we are writing the component for.

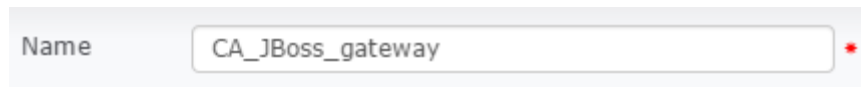
For the components and workflows, the names will be starting with “<CompanyName>_”. For parameters, this will become a suffix as shown “_<CompanyName> <parameterScope>”.

Components Names

The component name will start with “**CA_**” for CA technologies provided shared components, and then we will add the technology for which this component is being written. “**CA_***Componentname*”. For customers, this will be “**CompanyName_**”.

Name	CA_JBoss
------	----------

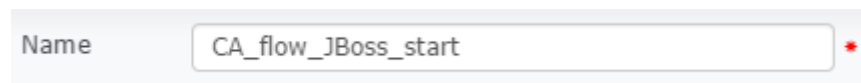
If additional sub-components are needed, the naming convention will be as follows: “**CA_***Componentname_subcomponent*”.



A screenshot of a web form with a label "Name" and a text input field containing the text "CA_JBoss_gateway". A red asterisk is visible to the right of the input field, indicating a required field.

Flow Names

Flows will be named as follows “**CA_flow_***Componentname_flowname*”.

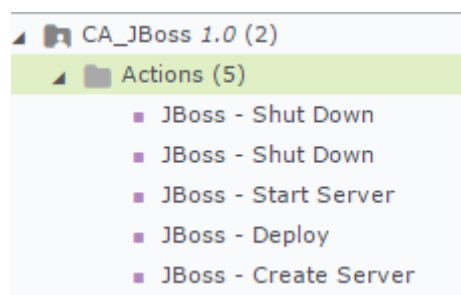


A screenshot of a web form with a label "Name" and a text input field containing the text "CA_flow_JBoss_start". A red asterisk is visible to the right of the input field, indicating a required field.

Actions Name

The actions that make up the shared components to be created by the CA team or other teams (these teams could be CA services on behalf of a customers, partners, or customers) need to retain their default names as much as possible; this is being done to avoid confusion by the support, pre-sales teams, partners, and customers that will be using these shared components libraries.

There is going to be a need to have the actions named based on what they are going to be accomplishing.



Parameters Names

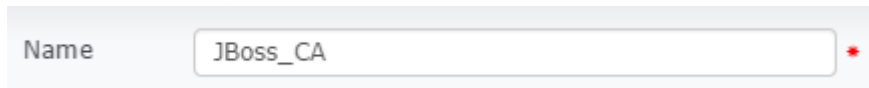
All parameters will have a suffix of “_**<CoName><parameterScope>**”; the scope will be based off the following scope values in Release Automation.

- Internal → no entry

- User Input → u
- Environment → e
- Release → r

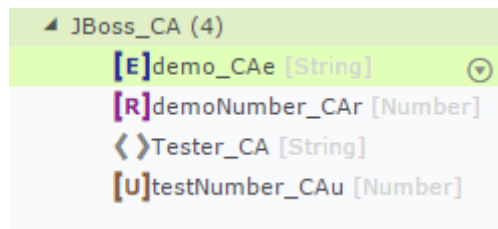
It is recommended that a parameters folder be created and include all the shared component's parameters in this newly created folder. This will help with maintainability issues.

The parameter folder needs to be named as "componentName_CA".



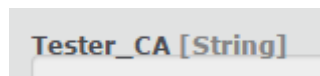
A screenshot of a form with a label 'Name' and a text input field containing the text 'JBoss_CA'. A small red asterisk is visible to the right of the input field.

A typical view of the component's parameter folder with multiple parameters added. Notice that the parameter type is already listed for each of the parameters in this particular folder.



Internal


Since the internal parameter will be the most common parameter being utilized, it needs to have a suffix as follows: "name_CA".



A screenshot of a parameter entry showing the text 'Tester_CA [String]'.

Environment

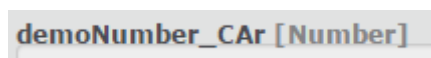
For environment parameter the naming suffix will be "name_CAe"



A screenshot of a parameter entry showing the text 'demo_CAe [String]'.

Release

For release parameter the naming suffix will be "name_CAR"



A screenshot of a parameter entry showing the text 'demoNumber_CAR [Number]'.

User Input

For user input parameter the naming suffix will be “name_CAu”

```
testNumber_CAu [Number]
```

Collection Names

Collections represent a special case, where collections can be defined under the following scopes: environment or release.

The naming suffix will follow this format:

Environment scope → “name_CAce”

```
colDemo_CAce [Collection Ele...
```

Release scope → “name_CAcR”

```
colTester_CAcR [Collection Ele...
```

Artifacts

The global artifacts that are part of the shared component, it is recommended that they be named as follows.

<sharedcomponent>artifact Type → jbossApps

Mapped Artifact Types

```
jbossApps  
jbossConfig
```

Exporting Shared Components

When creating an export file of the shared components, the following naming convention needs to be followed: “componentName_CA_vxxx”, where the ‘xxx’ represent the version number in Release Automation. It is recommended that the exported zipfile be password protected as well, the default password to be used would be “CARA_SharedComponent*”

It may be necessary that multiple shared components will need to be exported in a single zipfile package.

Documentation Conventions

The following documentation guide lines need to be followed.

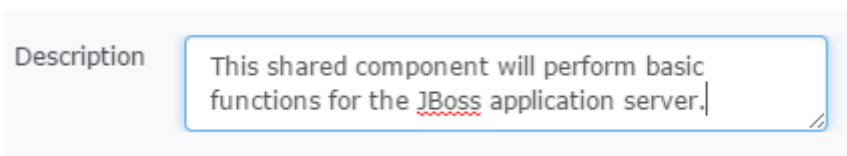
Release Doc

The release documentation will be based off a PDF format for portability purposes, at the same time the release document will also include a table of contents and be setup with the following sections:

- Introduction
- What's New
- Fixes
- Workflows
 - Listing of flows by component
- Environment variables
 - Generic
 - Listing by component
- Artifacts

Descriptions in Shared Components

In the description for the shared component, add a short description of what this shared component would be covering.



The flows need to be document for maintenance and usage. All actions added need to have a description of what the action will be doing within this given flow.

Description	<div>This flow will shutdown a JBoss application server, deploy a new service, then bring the</div>
-------------	---

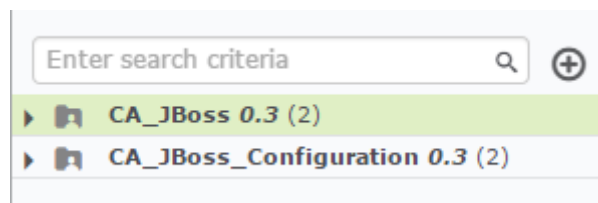
The description for the parameters needs to be completed describing what the parameter will be used for.

Best Practices

It is recommended that the following best practices be followed for the creation of the shared components. These practices will improve the quality and portability of the shared components.

Shared Components

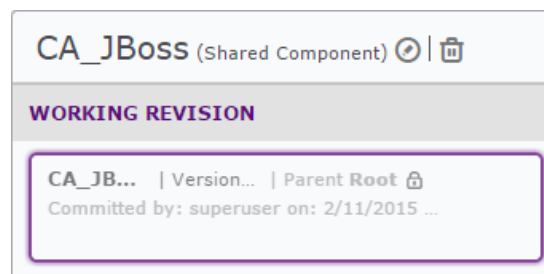
Part of the planning of the shared components that need to be created, it is important that you defined what type of work each of the components will be performing, below is a typical view, where you can see the default shared component for JBoss, which will mostly perform basic flows for deploying/removing applications to the JBoss application server. There is also another component designed to perform flows for the JBoss application server's configuration requirements.



Multiple Revisions

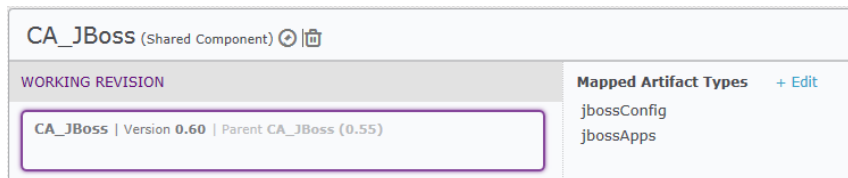
To avoid confusion after the creation of the distributable zipfile package, it is recommended that you export all the versions that you have been working on, afterwards remove all the older versions, and create a new distributable that contains just one revision (the latest version). You can then re-import the older versions for historical purposes.

That is one method to follow another method to follow, after the first release of your shared component is to include only the released revisions in upcoming shared components zipfile.



Mapped Artifact Types

The global artifact type that is part of the shared component will be created during the import phase, if the artifact types are not present already. You will need to add in the release doc for the shared component on the global artifact type that the shared component needs to use.

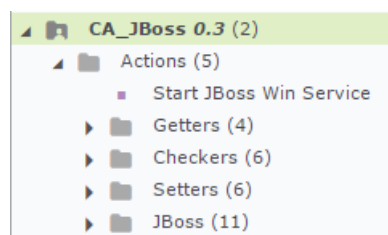


Shared Component's Actions

To improve readability of the actions that will make up a given shared component is recommended that categories be part of the actions, the following list of generic categories will provide you with a logical representation of the different actions and you can at a glance determine which action to use based on these generic categories.

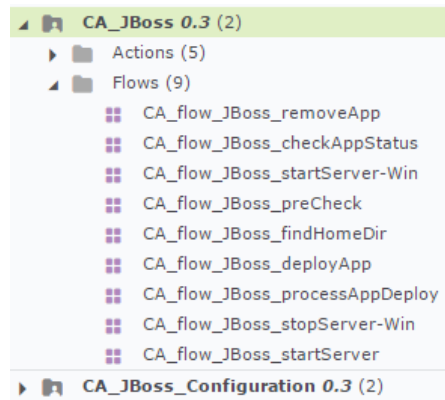
- Checkers
- Getters
- Setters
- <component's technology> → In this case, this will be i.e. JBoss

Below is a representation in how the generic categories will look like.



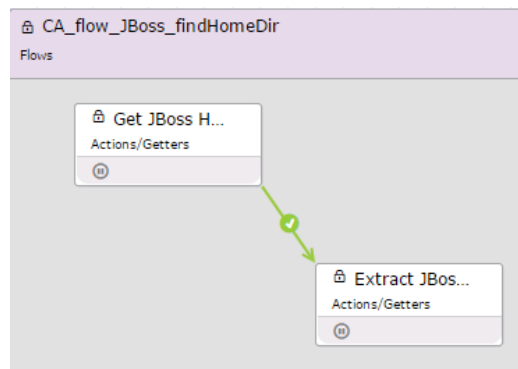
Shared Component's Flows

The flows need to be named and setup based on the work to be performed, this will reduce confusion, and provide the developers with a quick glance on the available flows.



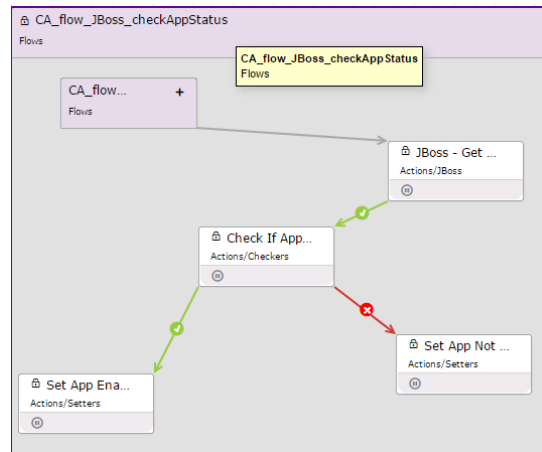
Generic Flows

Create generic flows that will be performing repetitive activities, for example in a JBoss installation is necessary to know the JBoss Home directory location, this is a necessary information prior to executing any JBoss actions. The results obtain by these generic flows needs to be placed in internal parameter variables as needed.



Number of actions in flows

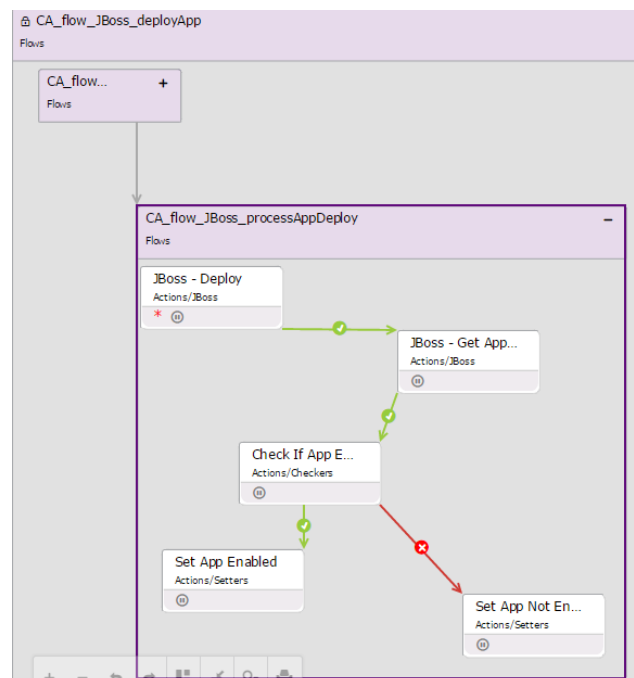
To reduce the complexity of the flows, it is important that the number of actions be limited. This can be accomplished by including other flows within a flow and keep the number of actions to a minimum. For example use four actions and another generic flow.



Loops in Flows

Loops will become very important when operating on global artifact types, therefore when you are creating a flow that will need to loop thru the global artifact type, it is recommended that the flow contains generic flows and another flow that will loop thru the global artifact type and perform the necessary actions.

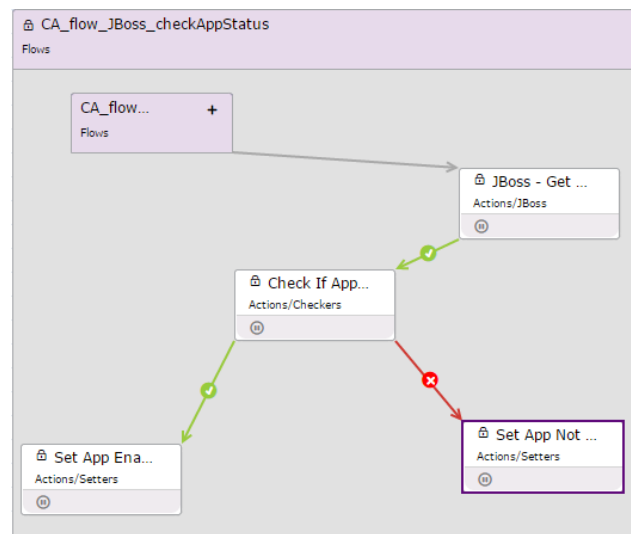
For artifact type, the loop type will be collections.



Setting Booleans as result of flow

As a best practice, it is recommended that you setup Boolean values for the different flows as a mechanism defining if the actions within the flow has completed successfully, or has failed.

For example in the image below, there is an action that checks the results of the status of the application in the JBoss application server, based on the this check, a Boolean value will be set accordingly and this value can be used to launch another flow as necessary.



Copyright Notice

Copyright © 2015 CA, Inc. All rights reserved. All marks used herein may belong to their respective companies. This document does not contain any warranties and is provided for informational purposes only. Any functionality descriptions may be unique to the customers depicted herein and actual product performance may vary.