Windchill Development Best Practice

**Name:** *Customizing Change Tables Designed for Workflow.*

<INTERNAL>

By: Saritha Mathai

Change History:

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Description** |
| 08/23/2107 | Mike McLachlan | Initial draft |
| 12/18/2017 | Bob Lach | Add advanced customization information |
|  |  |  |
|  |  |  |

</INTERNAL>

# Best Practice Name and Classification

## Name

Managing Impacted Baselines in a Change Process

## Classifications

Customization and configuration options available for handling updates to a baseline from a change workflow process. This includes association rules, constraints, workflow task templates, workflow expressions, custom action launch points, and potential error handling.

# Objective

## Problem Statement

As an administrator, I want to tailor a workflow process that has the capability to manage objects in a baseline through an approval process.

## Scope/Applicability/Assumptions

This document assumes familiarity with Association Rules, Workflow Processes, and Workflow Task Templates.

## Intended Outcome

At the end of this, the reader should have a good idea how to update baseline objects with a formal change process.

# Solution

There are several pieces to this capability that can be setup to help aid this process. They are association rules, association rule constraints, workflow task templates, and workflow templates.

## Prerequisite Knowledge

It is recommended to have an understand of the following:

* Association Rules
* Workflow Processes
* Workflow Task Templates

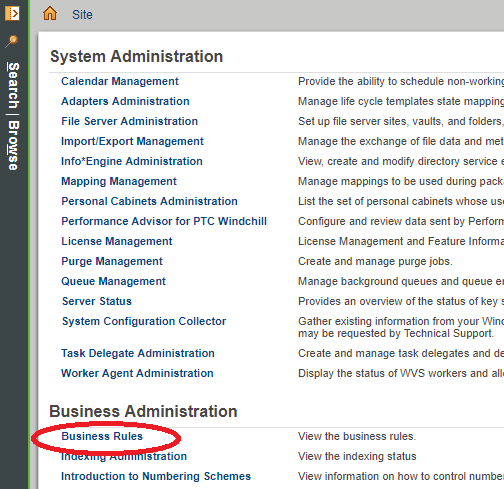
For advanced customization:

* Enumerated Type Customization Utility
* Java Code Development

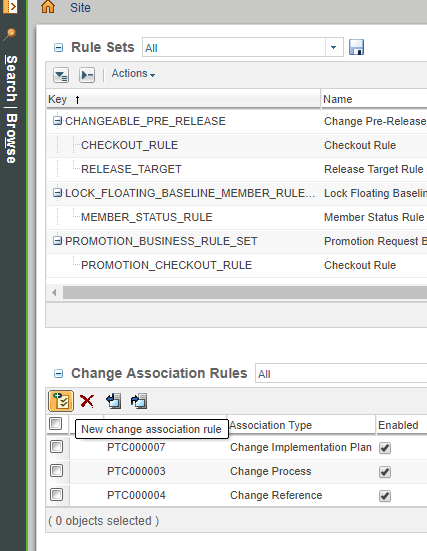
## Creating the Association Rules

If a workflow process of a change object drives the content of a baseline, then we must first be able to link the change object to the baseline to indicate the association. In order to do this, we must first define an association rule, of type Impact, where the Role A object is the Change object, and the Role B is the type of Baseline.

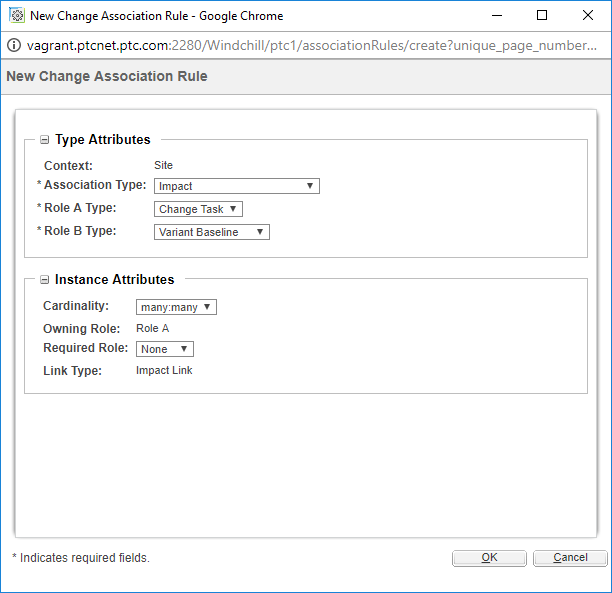
First navigate to the Site / Business Rules section.



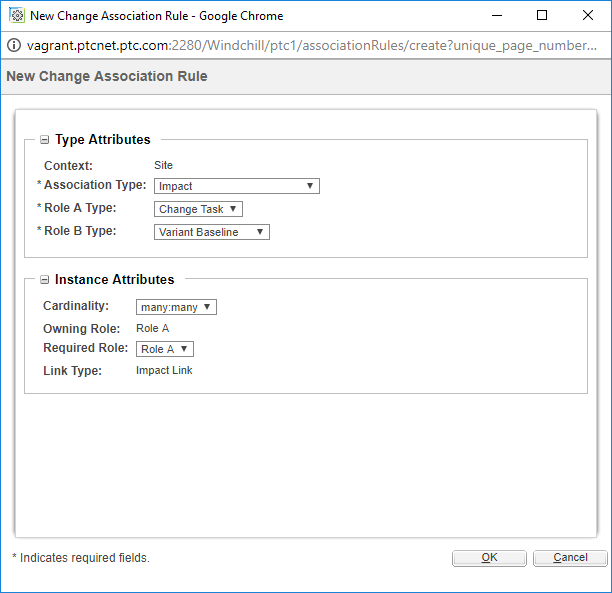
Invoke the action to create a new change association rule.



Choose Impact as the Association Type, and then choose relevant types for the Role A and Role B. For example, if you just want any Change Task to be able to manage any Variant Baseline, you would select:



Also note the Instance Attributes section. If you wanted to keep the system mostly as-is, and simply use the additional formal process when needed, the default options will likely suffice. However, let’s say for example, that you have a strict change process where all Change Tasks created must have an associated Variant Baseline. You could satisfy this requirement by switching the Required Role attribute to Role A.

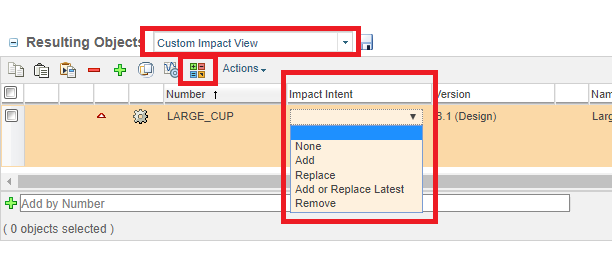


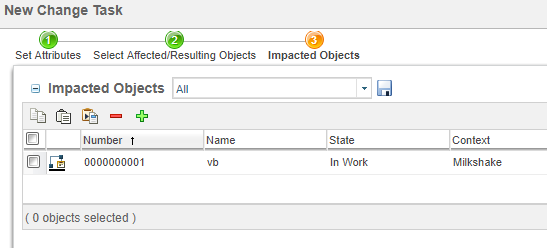
This concludes the necessary work for creation of the Association Rule.

## Exposing the Impact Intent

After the association rule is created, you will notice that when creating a Change Task it contains a 3rd step in the New Change Task wizard. This is for linking the Change Task to the Baseline. Additionally, if you observe the Resulting Objects table you will also notice a new action shows up at this point, and there is an additional configurable column you can add to the table called the Impact Intent. In order to use this properly, the column should be added to the table. This can be added to the Summary Table, Resulting Objects table, Complete Resulting Objects table, and the Review Resulting Objects table.

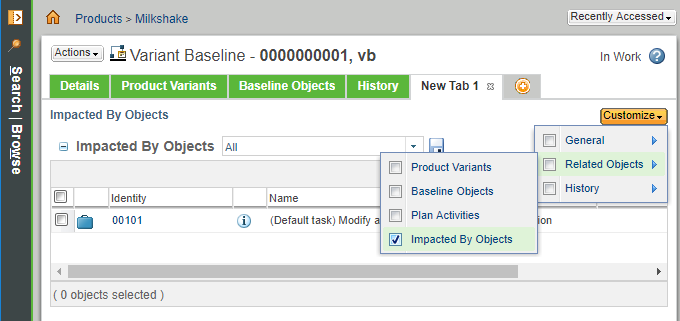
Example: Let’s say we wanted to add the part LARGE\_CUP to a Variant Baseline called VB. You would add the LARGE\_CUP as a Resulting Object, choose an Impact Intent of “Add” and then add the Variant Baseline VB to the Impacted Objects table.



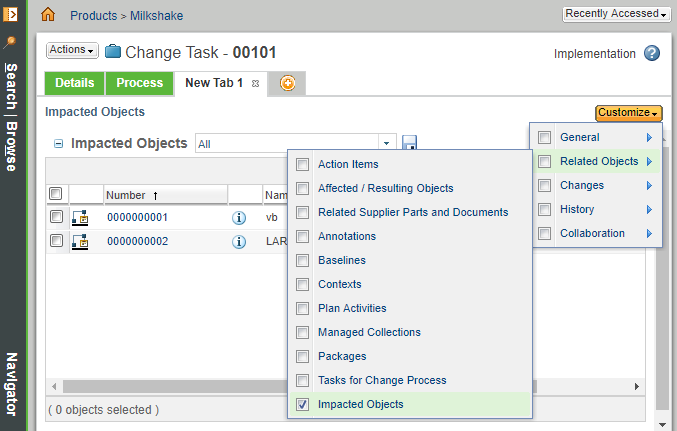


Now if we finish the wizard at this point, without doing anything else, we will notice that the LARGE\_CUP is not actually added to the Variant Baseline called VB yet. This is because the Default Workflow Template provided with Windchill does not invoke the service API to perform the updates. See section 3.4 on how to actually configure the workflow process to apply the actual updates.

There are a couple of additional tables to note at this point, which can help you understand what the related Impacted Objects are when viewing various objects. For example, if you’re viewing Variant Baseline information, and you want to know what Change Tasks are associated with it, you can add the Impacted By Objects table.



Likewise, you can configure the Impacted Objects table to show up on a Change Task.



This concludes the section of exposing the Impact Intent user interface components.

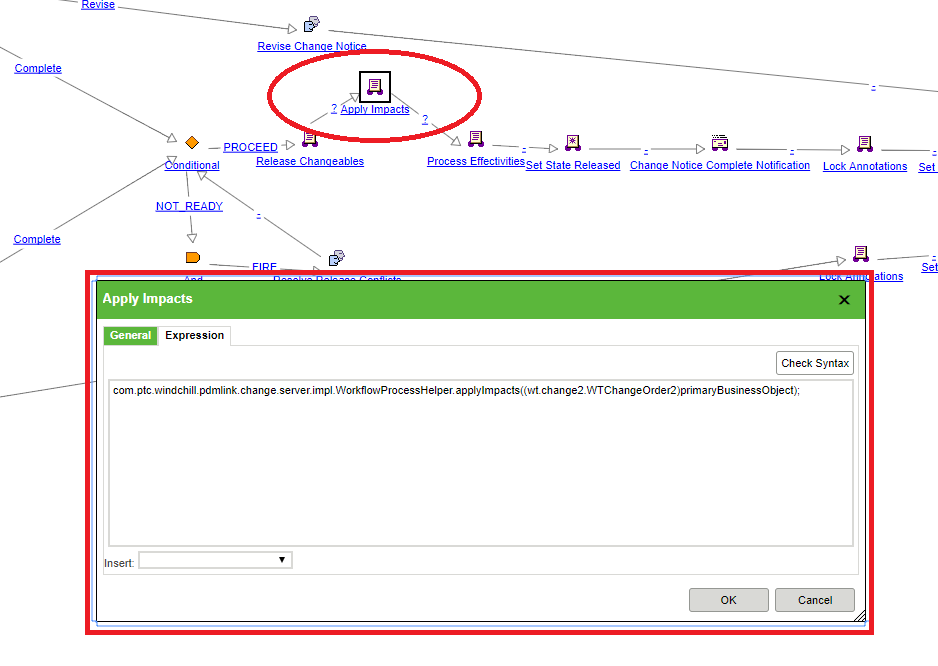
## Defining the Workflow Process

The Default workflow process must be updated in order to actually persist the updates against the baseline as indicated by the Impact Intent. An administrator currently has two options for where to place this configuration. It can either go in the Change Notice workflow or the Change Task workflow.

The Workflow Helper API that applies the impact is defined as follows:

|  |
| --- |
| /\*\*  \* Apply the impact of a change activity, and all the change activities of a change order,  \* on associated {@link wt.impact.Impacted} objects.  \*  \* <BR><BR><B>Supported API: </B>true  \*  \* @param changeItem ChangeActivity2 or ChangeOrder2 object, other types ignored.  \*  \* @return True if an associated Impacted object was updated. False if none were updated  \* or none are associated.  \*  \* @throws WTException  \*/  public static boolean applyImpacts(VersionableChangeItem changeItem) throws WTException |

Let’s say for example, that your business process was all-inclusive and if one Change Task should be rejected then they all would be. If that was the case, this would be an ideal opportunity to use the Change Notice workflow to apply impact, since it could apply the impacts to the baselines after all Change Tasks had completed successfully. You simply add a new workflow expression directly after the release of the Changeables to apply the impact.



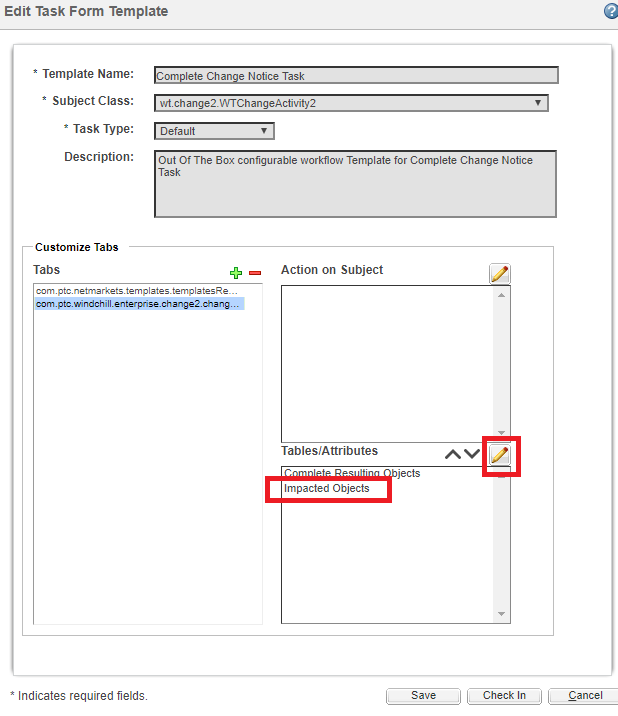
Be sure to save the updates to the workflow template. Note that workflows are iterated objects so you must do a check-in for new Change Notices to use the new workflow.

After doing this you should notice that updates to the baseline will occur after Audit Change Task is completed.

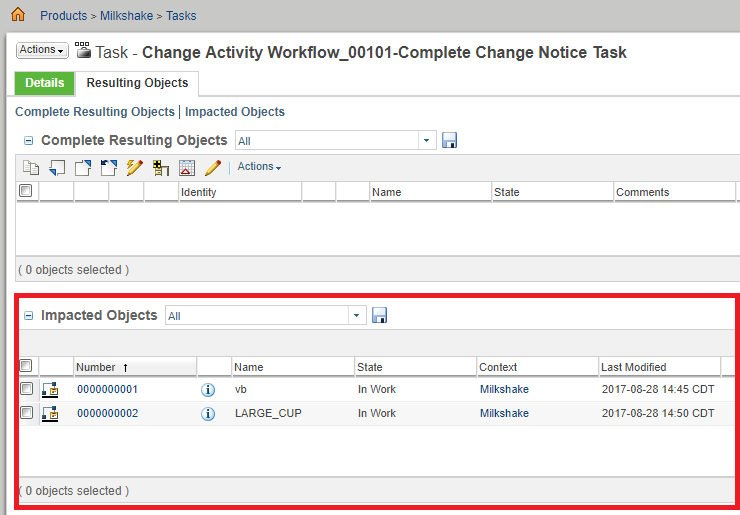
In the event of an error during a call to this API, a transaction will rollback all updates that were made, and an exception will be thrown from the API. If it is desired to have a rework loop, then the exception will need to be caught, and then it might be desirable to spawn off a new workflow task for analysis.

## Defining the Workflow Task Templates

Workflow task templates can be defined to include the Impacted Objects table. In order to configure them, you must go to the Site / Templates / Task Form Templates table. If for example, you wanted to add the Impacted Objects table to the Complete Change Notice Task you could Edit the existing Complete Change Notice Task and add it.



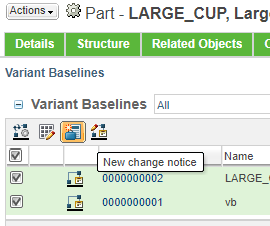
Then when viewing this task, it should be available in the workflow task page.



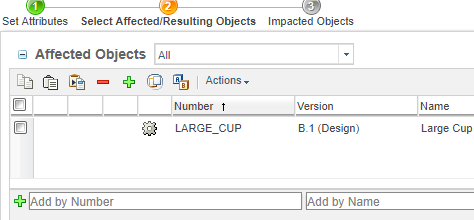
## Advanced Customization

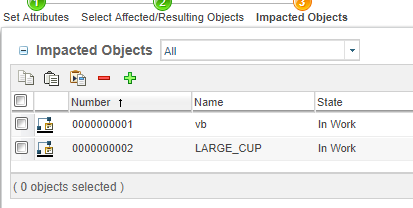
### Custom Launch Points for Creating Change Notices

When creating Change Notices, there are a number of ways to seed objects into the Impacting Objects table and the Affected Objects table. The primary locations where you can seed objects in are from the Variant Baseline table (as seen on the part info page) as well as from the Matrix Editor directly.



For example, in this case the VB chosen in the table was automatically seeded into the Impacted Objects table, and the part was seeded into the affected objects table.





If the default seed behavior is not sufficient, or if there is simply a desire to derive additional seeds from a different launch point, this capability is available, granted that they are valid Impact associations. The only thing the customizer must do is ensure that the hidden fields change\_selectedItems and initialImpactItems are populated with the relevant objects prior to the load of the wizard. These fields are already part of the Change Notice wizard. Additionally, there is already code in place to copy the hidden field into the Change Task wizard, so nothing needs to be done for that dealing with nested wizards.

If you view the source of the Change Notice wizard, you’ll observe that the object identifiers are passed like this and separated by the *#* character.

|  |
| --- |
| <input type="hidden" name="change\_selectedItems" id="change\_selectedItems" value="OR:wt.part.WTPart:131580#OR:wt.lotbaseline.LotBaseline:146122">  <input type="hidden" name="initialImpactItems" id="initialImpactItems" value="OR:wt.part.WTPart:131580#OR:wt.lotbaseline.LotBaseline:146122"> |

The customizer should follow this same format. Additional server side checks are done during persistence to ensure that the associations are valid, so the customization need not worry about persisting invalid associations.

There are two good examples of this, but there are many ways that this could happen. For JCA launch points, it is recommended to use a tag class. For GWT launch points, an action wrapper is recommended.

Tag example:

|  |
| --- |
| [String](http://bla-opengrok/s?defs=String&project=wnc-main) **[baselines](http://bla-opengrok/s?refs=changeItemClass&project=wnc-main)** = "wt.lotbaseline.LotBaseline:1234";  [String](http://bla-opengrok/s?defs=String&project=wnc-main) [**name**](http://bla-opengrok/s?refs=changeItemClass&project=wnc-main) = "initialImpactItems";  [response](http://bla-opengrok/s?defs=response&project=wnc-main).[append](http://bla-opengrok/s?defs=append&project=wnc-main)("<input type=\"hidden\" name=\"").[append](http://bla-opengrok/s?defs=append&project=wnc-main)([name](http://bla-opengrok/s?defs=name&project=wnc-main)).[append](http://bla-opengrok/s?defs=append&project=wnc-main)("\" id=\"");  [response](http://bla-opengrok/s?defs=response&project=wnc-main).[append](http://bla-opengrok/s?defs=append&project=wnc-main)([name](http://bla-opengrok/s?defs=name&project=wnc-main)).[append](http://bla-opengrok/s?defs=append&project=wnc-main)("\" value=\"");  [String](http://bla-opengrok/s?defs=String&project=wnc-main) [encodedValue](http://bla-opengrok/s?defs=encodedValue&project=wnc-main) = [HTMLEncoder](http://bla-opengrok/s?defs=HTMLEncoder&project=wnc-main).[encodeForHTMLAttribute](http://bla-opengrok/s?defs=encodeForHTMLAttribute&project=wnc-main)([baselines](http://bla-opengrok/s?defs=value&project=wnc-main));  [response](http://bla-opengrok/s?defs=response&project=wnc-main).[append](http://bla-opengrok/s?defs=append&project=wnc-main)([encodedValue](http://bla-opengrok/s?defs=encodedValue&project=wnc-main)).[append](http://bla-opengrok/s?defs=append&project=wnc-main)("\" >\n"); |

GWT Example:

|  |
| --- |
| [JSONArray](http://bla-opengrok/s?defs=JSONArray&project=wnc-main) [seeds](http://bla-opengrok/s?defs=seeds&project=wnc-main) = ([JSONArray](http://bla-opengrok/s?defs=JSONArray&project=wnc-main)) [result](http://bla-opengrok/s?defs=result&project=wnc-main).[get](http://bla-opengrok/s?defs=get&project=wnc-main)("soid");  **int** i = [seeds](http://bla-opengrok/s?defs=seeds&project=wnc-main).[size](http://bla-opengrok/s?defs=size&project=wnc-main)();  **for** ([Identifier](http://bla-opengrok/s?defs=Identifier&project=wnc-main) [baseline](http://bla-opengrok/s?defs=variant&project=wnc-main) : [selectedBaselines](http://bla-opengrok/s?defs=selectedVariants&project=wnc-main)) {  **if** ([baseline](http://bla-opengrok/s?defs=baseline&project=wnc-main) != [null](http://bla-opengrok/s?defs=null&project=wnc-main)) {  [seeds](http://bla-opengrok/s?defs=seeds&project=wnc-main).[set](http://bla-opengrok/s?defs=set&project=wnc-main)(i, **new** [JSONString](http://bla-opengrok/s?defs=JSONString&project=wnc-main)([EntityUtils](http://bla-opengrok/s?defs=EntityUtils&project=wnc-main).[toOID](http://bla-opengrok/s?defs=toOID&project=wnc-main)([baseline](http://bla-opengrok/s?defs=baseline&project=wnc-main))));  i++;  }  }  **if** ([**log**](http://bla-opengrok/xref/wnc-main/wcmod/modules/ProductFamilyManagementUI/src/com/ptc/windchill/enterprise/productfamily/pfsb/client/action/CreateChangeNoticeGWTAction.java#log).[isDebugEnabled](http://bla-opengrok/s?defs=isDebugEnabled&project=wnc-main)()) {  [**log**](http://bla-opengrok/xref/wnc-main/wcmod/modules/ProductFamilyManagementUI/src/com/ptc/windchill/enterprise/productfamily/pfsb/client/action/CreateChangeNoticeGWTAction.java#log).[debug](http://bla-opengrok/s?defs=debug&project=wnc-main)("initialImpactItems appended the selected oids and the items in selectedBaselines: " + [seeds](http://bla-opengrok/s?defs=seeds&project=wnc-main));  }  [result](http://bla-opengrok/s?defs=result&project=wnc-main).[put](http://bla-opengrok/s?defs=put&project=wnc-main)("initialImpactItems", [seeds](http://bla-opengrok/s?defs=seeds&project=wnc-main)); |

### Custom Impact Intent Values

Windchill provides a standard set of Impact Intent values to control baseline changes. These standard values can be disabled in the user interface, but should not be removed or redefined.

1. None – no change to impacted baselines (default if no Impact Intent value is selected).
2. Add – add object to impacted baselines that don’t have it.
3. Replace – replace object in impacted baselines that have an earlier or later version.
4. Add or Replace Latest – add object to impacted baselines that don’t have it, or replace object in impacted baselines that have an earlier version (i.e., do not replace a later version).
5. Remove – remove object from impacted baselines that have it.

New Impact Intent values can be added using the Enumerated Type Customization utility. Instructions for using this utility can be found in the Windchill Help Center.

The default Impact Intent value, if specified, will be used when no other value is selected. The standard Windchill default value is “None”, but may be changed to another value with the Enumerated Type Customization utility.

The order of Impact Intent values shown in the Enumerated Type Customization utility is not important. The actual order shown in the Windchill user interface is controlled by the Change Task impactor delegate (see next section).

### Impact Delegates

Java delegate classes are used to control how a Change Task can impact Variant (Lot) baselines and Managed baselines:

Delegates ChangeActivity2ImpactorDelegate, ManagedBaselineImpactedDelegate and LotBaselineImpactedDelegate can be extended to provide custom control of the baseline management process.

<<abstract>>

*ImpactDelegate*

<<abstract>>

*ImpactorDelegate*

ChangeActivity2ImpactorDelegate

ManagedBaselineImpactedDelegate

LotBaselineImpactedDelegate

<<abstract>>

*ImpactedDelegate*

### ChangeActivity2ImpactorDelegate

Java delegate class wt.change2.ChangeActivity2ImpactorDelegate controls how a Change Task participates in the baseline management process. This delegate may be extended to provide custom processing features. To create a custom delegate start with the following code:

public class CustomChangeActivity2ImpactorDelegate extends ChangeActivity2ImpactorDelegate {

private static final long serialVersionUID = 1L;

public CustomChangeActivity2ImpactorDelegate() {

super();

}

public CustomChangeActivity2ImpactorDelegate(Impactor impactor) {

super(impactor);

}

}

A variety of supported delegate methods may then be overridden to tailor the code for the desired business process. For example, the following methods may be overridden:

* List<ImpactIntentType> getAllowedIntents() – controls the list of Impact Intent values shown in the Change Task’s Resulting Objects table. Override this method to add custom values or remove unwanted standard values. The list also specifies the order in which the Impact Intent values are displayed in the selection menu.
* WTKeyedMap getImpactIntents() – returns a map of Resulting Objects to the Impact Intent values selected by the user. The standard implementation returns all Resulting Objects in the Change Task. Override this method to remove object types if the business process dictates that some object types should never be added to a baseline, or if some object types should have different default intent values than others.

For complete descriptions of supported methods see the JavaDoc for classes wt.impact.ImpactDelegate, wt.impact.ImpactorDelegate, and wt.change2.ChangeActivity2ImpactorDelegate.

### LotBaselineImpactedDelegate / ManagedBaselineImpactedDelegate

Java delegate classes wt.lotbasline.LotBaselineImpactedDelegate and wt.vc.baseline.ManagedBaselineImpactedDelegate control how a Variant baseline and a Managed baseline participate in the baseline management process, respectively. These delegates may be extended to provide custom processing features. To create a custom Variant baseline delegate start with the following code:

public class CustomLotBaselineImpactedDelegate extends LotBaselineImpactedDelegate {

private static final long serialVersionUID = 1L;

public CustomLotBaselineImpactedDelegate() {

super();

}

public CustomLotBaselineImpactedDelegate(Impactor impactor) {

super(impactor);

}

public CustomLotBaselineImpactedDelegate(WTCollection impacteds) {

super(impacteds);

}

}

Use similar code to start a CustomManagedBaselineImpactedDelegate class, but extend it from ManagedBaselineImpactedDelegate instead.

A variety of supported delegate methods may then be overridden to tailor the code for the desired business process. For example, the following methods may be overridden:

* List<ImpactIntentType> getIntentApplyOrder() – controls which Impact Intent values are supported by the delegate, and the order in which they are processed. Override this method to add custom values or remove unwanted standard values.
* boolean applyImpactIntent(ImpactIntentType intent, Impacted impacted, WTSet impactObjs) – apply an Impact Intent on a baseline with a set Resulting Objects. The baseline is passed in as the impacted parameter, and the Resulting Objects are passed in as the impactObjs set. Override this method to support a custom Impact Intent value. Note that custom implementations should always call the super method for Impact Intent values it does not implement itself.  
    
  For example, the following code supports new Impact Intent values “Add In-Procurement”, “Add In-Assembly” and “Substitute”. It also provides a new implementation for the standard “Add”

@Override

protected boolean applyImpactIntent(ImpactIntentType intent, Impacted impacted,   
 WTSet impactingObjs) throws WTException {

if ( !(impacted instanceof LotBaseline) ) {

return false;

}

LotBaseline lotBaseline = (LotBaseline) impacted;

if ( ImpactIntentType.ADD.equals(intent) ) {

return applyAddImpactWithStatus(lotBaseline, impactingObjs, "InWork");

}

else if ( CustomImpactHelper.ADD\_IN\_PROCUREMENT.equals(intent) ) {

return applyAddImpactWithStatus(lotBaseline, impactingObjs, "InProcurement");

}

else if ( CustomImpactHelper.ADD\_IN\_ASSEMBLY.equals(intent) ) {

return applyAddImpactWithStatus(lotBaseline, impactingObjs, "InAssembly");

}

else if ( CustomImpactHelper.SUBSTITUTE.equals(intent) ) {

return applySubstituteImpact(lotBaseline, impactingObjs);

}

return super.applyImpactIntent(intent, lotBaseline, impactingObjs);

}

protected boolean applyAddImpactWithStatus(LotBaseline lotBaseline, WTSet addedObjs,

String status) throws WTException {

:

// Call super method applyAddImpact(lotBaseline, addedObjs)then

// set the status value on the new baseline member link.

:

}

protected boolean applySubstituteImpact(LotBaseline lotBaseline,

WTSet substitutedObjs) throws WTException

:

// Implement the "Substitute" intent.

:

}

For complete descriptions of supported methods see the JavaDoc for classes wt.impact.ImpactDelegate, wt.impact.ImpactedDelegate, wt.lotbaseline.LotBaselineImpactedDelegate and wt.vc.baseline.ManagedBaselineImpactedDelegate.

### Register Custom Delegates

Impact delegates are registered in the /Windchill/codebase/service.properties file using the Windchill xconfmanager utility. Note that an integer “selector” value is given to specify the order in which multiple delegates are applied. If a selector value is not given, or if it is not an integer, an informational message will be added to the method server log each time the delegate is instantiated.

When multiple delegates are involved in a single “apply impacts” action they will be sorted into groups having the same selector value. The groups will then be applied in increasing selector value order, with all ImpactorDelegate groups applied to each ImpactedDelegate group in turn. Multiple delegates within a single group will be applied in random order. If a selector value is not given, or if it is not an integer, then the delegate will be assigned to a group that is applied last.

For example, to register three custom delegates in place of the standard delegates create a file named RegisterCustomDelegates.xconf with the contents shown below. Replace the example serviceClass names shown below with the actual fully qualified custom delegate class names used for the customization.

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE Configuration SYSTEM "xconf.dtd">

<Configuration targetFile="codebase/service.properties">

<Service context="default" name="wt.impact.ImpactorDelegate">

<Option selector="10" cardinality="duplicate"

serviceClass="custom.CustomChangeActivity2ImpactorDelegate"

requestor="wt.change2.ChangeActivity2"/>

</Service>

<Service context="default" name="wt.impact.ImpactedDelegate">

<Option selector="10" cardinality="duplicate"

serviceClass="custom.CustomManagedBaselineImpactedDelegate"

requestor="wt.vc.baseline.ManagedBaseline"/>

</Service>

<Service context="default" name="wt.impact.ImpactedDelegate">

<Option selector="20" cardinality="duplicate"

serviceClass="custom.CustomLotBaselineImpactedDelegate"

requestor="wt.lotbaseline.LotBaseline"/>

</Service>

</Configuration>

Edit file $WT\_HOME/declarations.xconf and add the following line near the end of the file (i.e., just above the line containing </Configuration>). Replace the path and file name shown below with the actual path and file name used for the customization.

<ConfigurationRef xlink:href="codebase/custom/RegisterCustomDelegates.xconf"/>

In a Windchill shell run command "xconfmanager –pF" and then restart the method server(s).