Name: Change Workflow Closure Best Practice

<INTERNAL>

Change History:

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Description** |
| 8/13/2015 | Nathan Kral | Initial Draft |
| 8/17/2015 | Nathan Kral | Incorporated Review Comments |

</INTERNAL>

</INTERNAL>

# Best Practice Name and Classification

## Name

*Change Workflow Closure* Best Practice

## Classifications

Core Design Concept

## Also Known As

None

</INTERNAL>

# Objective

## Problem Statement

Problem: How to support closure for change object associations using flexible change process associations.

## Background

In prior releases change object associations were prescriptive in that problem reports can associate to a change request and change requests to a change notice. The supported API’s for closing the change objects are also specific to the supported relationships. Flexible change process associations now allow for configurable change object associations. Updated APIs are now available to fully support flexible change process associations.

A change object is considered closed when the resolution date and resolved state (i.e. Canceled or Resolved) are set on a change object. The two ways to close change objects are forced and synchronized closure. Synchronized closure is when a change object listens for an event and checks to see if it can close itself. Forced closure is when a different change object’s workflow process closes another related change object, forcing the state and resolution date to be set. For most cases synchronized closure is a better practice as this technique avoids dead lock and stale object issues that can sometimes occur when using forced closure for change objects having their own workflow process.

## Scope/Applicability/Assumptions

The scope of this best practice is to outline recommendations for supporting closure of change object associations using flexible change links.

## Intended Outcome

The intended outcome of this best practice is to help provide guidance on how to configure workflows to work with flexible change process associations.

# Solution

## Solution Statement

Solution: New APIs and Change Events are available to support closure of Change Objects associations using flexible links.

## Prerequisite knowledge

To apply this best practice, you need to have an understanding of the following:

* Basic development involving JAVA
* Defining workflow expressions

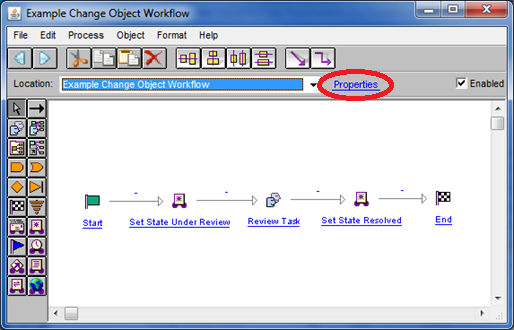
The [See Also](#_See_Also) section below includes references to many or all of these subjects.

## Solution Elements

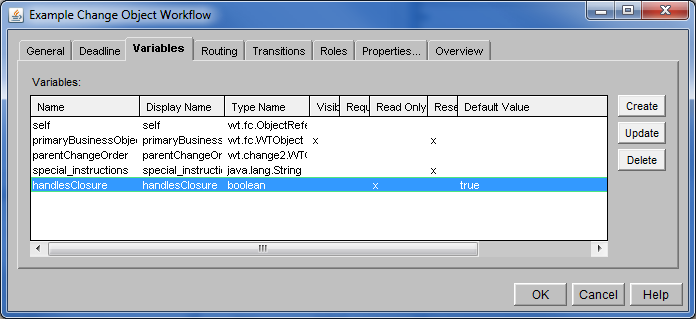
|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| com.ptc.windchill.pdmlink.change.server.impl. WorkflowProcessHelper | Class | This class contains static methods to support the change management workflow processes. The static methods in this class are intended to be called by routing expressions and synchronization robot expressions. |
| wt.change2.ChangeProcessLink | Class | The flexible change association link class. |
| wt.change2.AddressedBy | Class | Deprecated association link for Change Request to Change Notice. |
| wt.change2.FormalizedBy2 | Class | Deprecated association link for Problem Reports or Variances to Change Request. |
| handlesClosure | boolean | A Boolean property that can be specified as a workflow template property variable to specify if a change object workflow process is able to close itself. |
| CHILD\_CHANGE\_OBJECT\_STATE\_CHANGE | ChangeService2Event | The state was changed on child change object. The event target will contain the parent change object and have the child as a secondary target |
| PARENT\_CHANGE\_OBJECT\_STATE\_CHANGE | ChangeService2Event | The state was changed on parent change object. The event target will contain the child change object and have the parent as a secondary target. |
| isRelatedParentsInStates | WorkflowProcessHelper API | Looks up the related parent change objects and determines if each one is in one of the given states. |
| isRelatedChildrenInStates | WorkflowProcessHelper API | Looks up the related child change objects and determines if each one is in one of the given states. |
| setResolutionDate | WorkflowProcessHelper API | This method sets the resolution date of a change object. It does not set the resolution state. |
| closeChangeChildren | WorkflowProcessHelper API | Looks up the children of the given change object and closes them with the given states. |
| closeChangeParents | WorkflowProcessHelper API | Looks up the parents of the given change object and closes them with the given states. |

## Procedure – Update Workflow to use Synchronized Closure

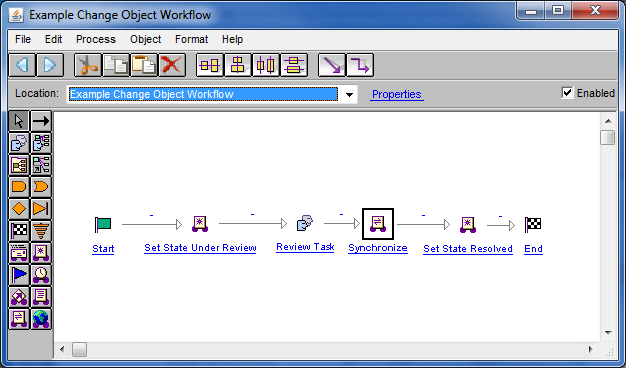
The following example will start with a basic workflow that goes from Under Review to Resolved state after a review task is completed with synchronized closure.



### Select the workflow template Properties link. Go to the Variables tab and add a new boolean variable named *handlesClosure* with the default value set to *true*.



### Add a new Synchronize expression between Review Task and Set State Resolved.



### Update the new Synchronize expression to synchronize on an “Object Event”. Select either the “PARENT CHANGE OBJECT STATE CHANGE” or “CHILD CHANGE OBJECT STATE CHANGE” event. See [Solution Elements](#_Solution_Elements) for descriptions of each event. Add one of the following two expressions in the Initial Expression and Routing Expression:

When listening for “PARENT CHANGE OBJECT STATE CHANGE”:

if(com.ptc.windchill.pdmlink.change.server.impl.WorkflowProcessHelper.*isRelatedParentsInStates* (primaryBusinessObject, new String[]{"*RESOLVED*"})) {

result = "Resolved";

}

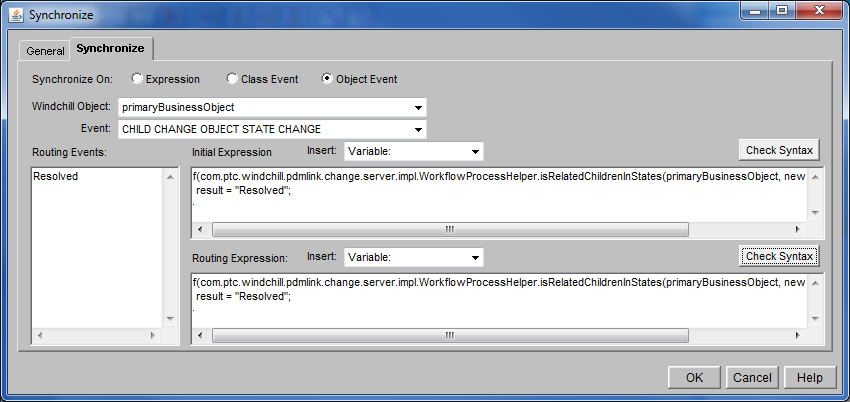
When listening for “CHILD CHANGE OBJECT STATE CHANGE”:

if(com.ptc.windchill.pdmlink.change.server.impl.WorkflowProcessHelper.*isRelatedChildrenInStates*(primaryBusinessObject, new String[]{"*RESOLVED*"})) {

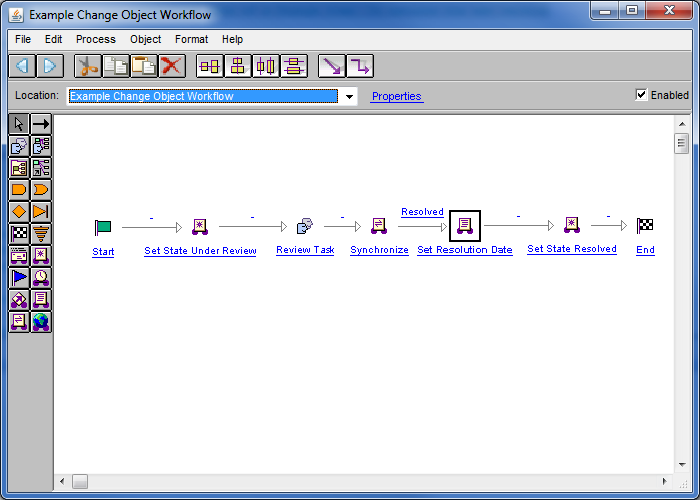
result = "Resolved";

}

Note that the role A of the change object relationship is considered to be the parent and the role B is the child object.

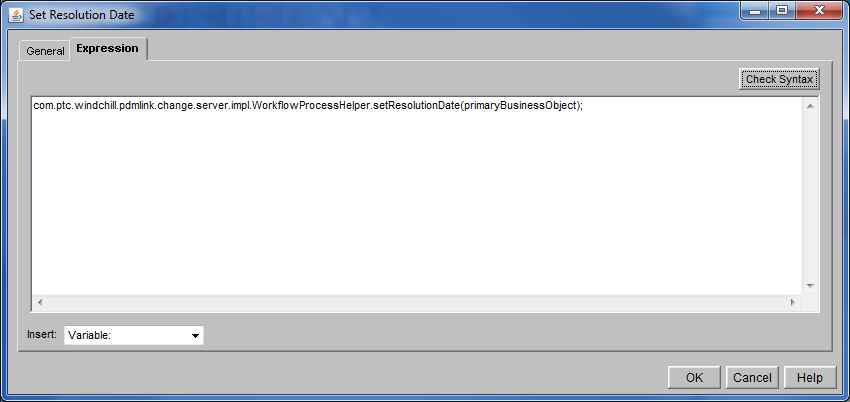


### Add a new Expression between Synchronize and Set State Resolved. Set its name to “Set Resolution Date”.



### Update “Set Resolution Date” to set the resolution date on the change object.

com.ptc.windchill.pdmlink.change.server.impl.WorkflowProcessHelper.setResolutionDate(primaryBusinessObject);

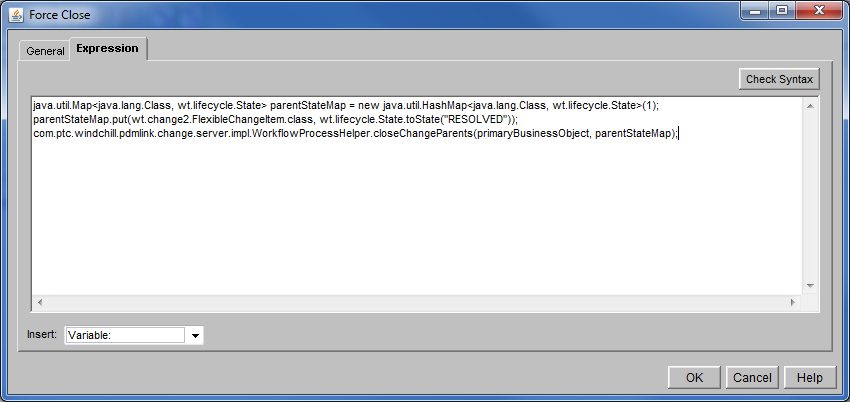


### If force closure is required for the change process a new Expression can be added after “Set State Resolved” to close either the parent or child associations. The following is an example expression implementation which forces the associated parent change objects to close in the Resolved state.

java.util.Map<java.lang.Class, wt.lifecycle.State> parentStateMap = new java.util.HashMap<java.lang.Class, wt.lifecycle.State>(1);

parentStateMap.put(wt.change2.FlexibleChangeItem.class, wt.lifecycle.State.toState("*RESOLVED*"));

com.ptc.windchill.pdmlink.change.server.impl.WorkflowProcessHelper.*closeChangeParents*(primaryBusinessObject, parentStateMap);



# Deprecated Link Support

The deprecated wt.change2.AddressedBy2 and wt.change2.FormalizedBy links are also supported by the new change workflow closure APIs and events. Any custom workflow processes can be updated prior to using Flexible change process links. The provided example change workflow process templates support both the new change process associations and the deprecated associations.

Review the following table in order to determine which APIs and events to use for the deprecated links:

|  |  |  |
| --- | --- | --- |
| **Deprecated Link** | **Parent** | **Child** |
| wt.change2.AddressedBy2 | Change Request | Change Notice |
| wt.change2.FormalizedBy | Problem Report or Variance | Change Request |

# Limitations

## Old and New Workflow Processes

In some cases , old workflow processes can be configured to work with new workflow processes. For example, Problem Reports or Variances that complete their workflow process after being approved can still work with a new Change Request workflow, but the Change Request workflow process will still need to call the force close on the parent associations. If the Problem Reports or Variances with a new workflow processes are also associated they will not be forced to closed when the *handlesClosure* is set to *true*. The Change Request workflow process template provided in 11.0 is configured to support both the previous release and new Problem Report and Variance workflow processes.

Using the force closure API on existing associated change object processes that already have synchronized closure will likely cause unexpected results. For example, the Change Request workflow process already has a synchronized closure for when the Change Notice is resolved. If the Change Notice workflow process was updated to force close the parent associations, then additional sequences in the Change Request process such as locking annotations would not happen.

## Required Updates to the Change Request Workflow Process

The Change Request workflow process must be updated in order for it to work with the flexible change process associations. The ChangeService2Event.CN\_STATE\_CHANGED is no longer supported with flexible change process associations. The event will still work for the deprecated wt.change2.AddressedBy2 associations.

## Closure with Multiple Associations

Direction of closure should be considered when determining which change object types are defined as the parent (role A) and child (role B) role of the flexible change process association. Note that the owning role can be defined separately from the parent or child role. For example, given the following relationships:

Problem Reports associate to Change Requests

Change Request associate to Change Notices

The recommendation is to use the direction of closure when defining the association rules. Defining the association rule in the direction of closure guarantees that the Change Request waits for all associated Change Notices to close and a Problem Report waits for all associated Change Requests to close. If the Change Request is considered to be the owner role for both associations then consider defining the change process association rules with the roles defined in the direction of closure as follows:

|  |  |  |
| --- | --- | --- |
| Role A (Parent) | Role B (Child) | Owner Role |
| Problem Report | Change Request | Role B |
| Change Request | Change Notice | Role A |

Additional considerations are required if the rules are not defined in the direction of closure. Consider the following changes to the rule definitions:

|  |  |  |
| --- | --- | --- |
| Role A (Parent) | Role B (Child) | Owner Role |
| *Change Request* | *Problem Report* | *Role A* |
| Change Request | Change Notice | Role A |

If the general isRelatedChildrenInStates API is used in the Change Request workflow process without specifying a specific change object type it would result in the Change Request waiting for the Problem Reports to close. The Problem Report workflow process is also waiting for the Change Request to close resulting in a closure deadlock. In order prevent closure deadlock the specific change object types should be specified when using the isRelatedChildrenInStates API. In the Change Request workflow process the Change Notice type should be specified so that the associated Problem Reports are ignored.

# Sample Code

## Examples of Usage in Windchill Code

Example change object workflow templates are provided and can be found in the Site>Utilities>Workflow Template Administration

# See Also

<INTERNAL>

## Related Best Practices

</INTERNAL>

## Related Package/Class Javadoc

com.ptc.windchill.pdmlink.change.server.impl.WorkflowProcessHelper

## Related Customization Documentation

## Other Related Windchill Documentation

## Related Websites

## Other

<INTERNAL>

## Internal Windchill Dev Resources

</INTERNAL>

# <INTERNAL>Other Windchill Development Considerations

None

</INTERNAL>