Windchill Development Best Practice

Name: *Replicated Objects as Change Task Resulting Objects*

<INTERNAL>

Change History:

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Description** |
| 4/25/2013 | Bob Lach | Initial version. |

# Best Practice Name and Classification

## Name

Replicated Objects as Change Task Resulting Objects

## Classifications

Low level – replication, customization.

</INTERNAL>

# Objective

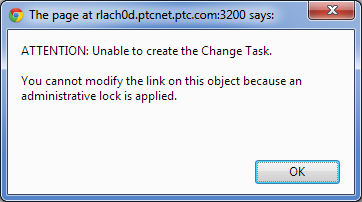
## Problem Statement

Add a replicated object as resulting object of a change task, and optionally assign it an effectivity value.

## Background

The term “replicated object” defines a non-modifiable object that is copied, or replicated, from a different Windchill installation. They are created by importing a Zip file with the Received Delivery Management utility. Replicated objects are subject to a special mechanism called an “administrative lock” that prevents them from being modified on the importing system.

Under normal circumstances a replicated object could not be added as a resulting object to a change task because the change process would not be able to update its life cycle state. If such an attempt is made then the following error message is given when the change notice is completed:



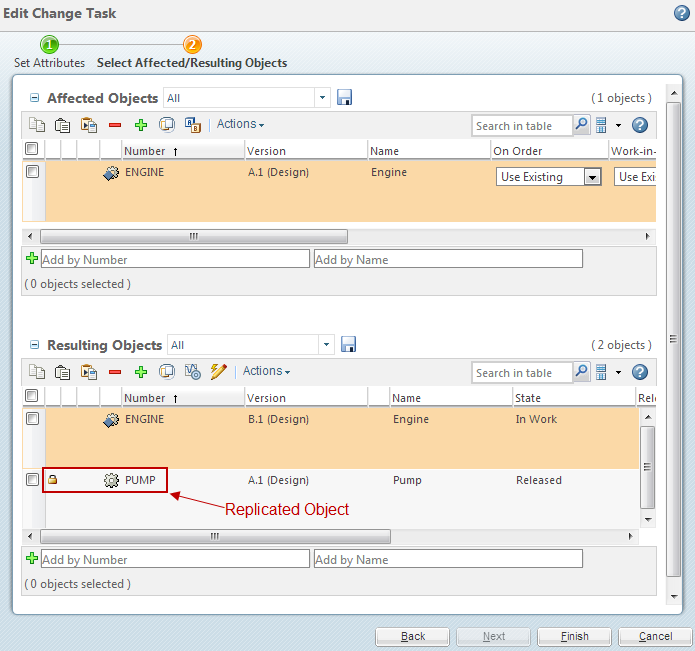
It is required in some circumstances, however, to be able to add a replicated object that has already been released to a change task (e.g., as a new component of an assembly). It may also be required to add an effectivity value to the replicated object when it is used a resulting object.

## Scope/Applicability/Assumptions

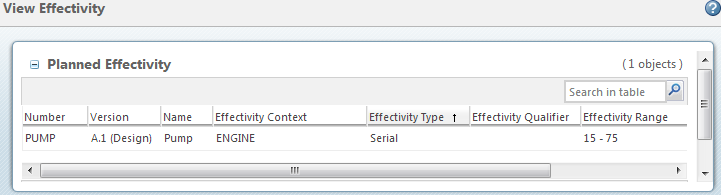
This document assumes familiarity with creating change tasks, adding resulting objects, and assigning effectivity values.

## Intended Outcome

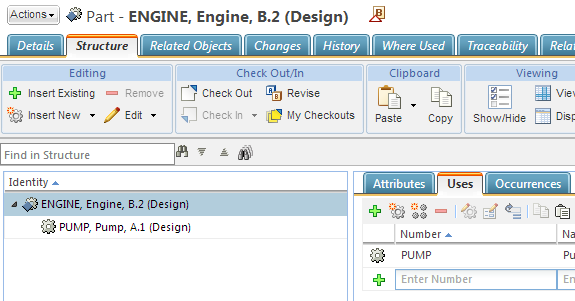
A resulting object is added to a change task as a resulting object. In the example below, Pump is a replicated object that is added to the Engine change task as a resulting object. Note that the padlock glyph indicates that Pump has an administrative lock and is common to all replicated objects.



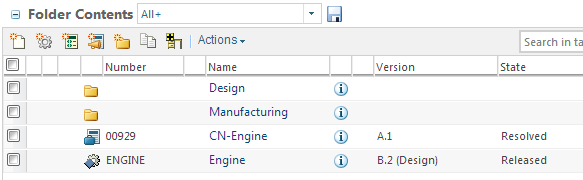
Further, a serial effectivity has been added Pump.



To implement the change Pump is added to the Engine structure.



When the change notice is completed the Engine is advanced to a Released state. The Pump, which was already in a Released state, is not changed.



# Solution

## Solution Statement

Customize Windchill to allow replicated objects as change task resulting objects, and prevent the change process from attempting to promote replicated objects to a Released state.

## Prerequisite knowledge

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

* Setting a system property
* Creating a custom life cycle template
* Creating an import life cycle mapping

# Customization Points

The customization steps in this section should be completed prior to importing replicated objects that are to be used as change task resulting objects. If they are completed after the import is complete then you can use the “Reassign Life Cycles” action to reassign the life cycle template of a replicated object to the custom one created in 4.2.

## Set System Property for Replicated Resulting Objects

A system property allows a replicated object to be added as a change task resulting object.

Perform the following steps:

1. Add the following line to $WT\_HOME/site.xconf:  
     
   <Property name="wt.fc.adminlock.AdministrativeLockType.change2.ProductDesignPkg" overridable="true" targetFile="codebase/wt.properties" value="RESULTING\_OBJECTS"/>
2. Run this command in a Windchill shell: **xconfmanager –pF**
3. Restart the method server(s).

## Create Custom Life Cycle Template for Replicated Objects

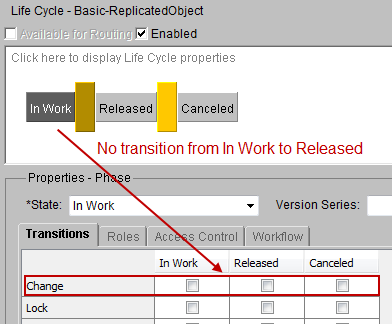
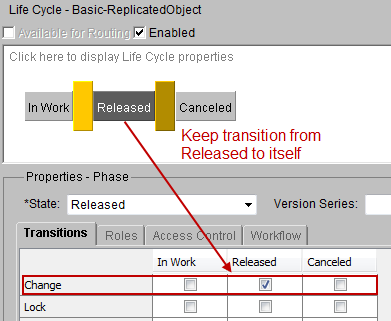
The out of the box (OOTB) Windchill change notice workflow includes a step to promote resulting objects to a Released state. This should not be attempted for replicated objects, however, because their life cycle state is controlled by a different Windchill installation. To prevent this you can create special life cycle templates for replicated objects that do not include change transitions from non-Released to Released states. The only change transition such templates should include is from the Released state back to itself.

For example, the OOTB Basic workflow includes these change transitions:

1. In Work 🡪 Released
2. Released 🡪 Released

Replicated objects that use the Basic life cycle template should not include transition “In Work 🡪 Released”, so a new template that does not include this transition should be created.

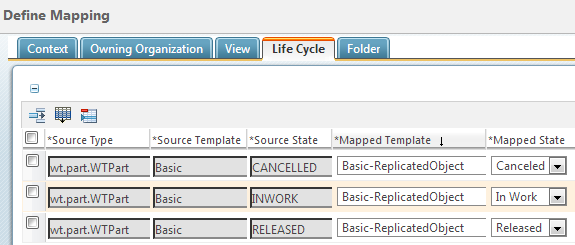
Perform the following steps:

1. Open the Life Cycle Template Administration utility and use the “Save As” action to make a copy of an existing template that you want to customize for replicated objects. For example, copy the Basic life cycle to a new one named “Basic-ReplicatedObject”.
2. Edit the copied template and remove all change transitions from non-Released to Released states, and keep only a change transition from the Released to Released. For example:  
     
     
     
   
3. Save and Check In the custom life cycle template.
4. Repeat this process for all life cycle templates that will be used for replicated objects.

## Create Import Mappings for Replicated Object Life Cycles

When replicated objects are imported they are normally assigned to the life cycle template that is specified in the exported Zip file. In this case, however, they should be assigned to the custom life cycle templates described in 4.2. This is accomplished by creating life cycle import mappings prior to importing the objects.

Perform the following steps:

1. Open the Mapping Management utility and select the “Define Mapping” action for the exporting Windchill installation.
2. Select the “Life Cycle” tab from the Define Mapping window and add rows as needed to map all states from the exported life cycle template to the custom template. For example, the OOTB Basic life cycle has three states: In Work, Released and Canceled. So three rows are needed for mapping.
3. Fill in the mapping fields so each source state maps to its associated target state. Note that this utility defines “Source” values as those coming from the exporting Windchill installation and “Mapped” values as those defined in the importing Windchill installation. Also note that the “Source” values are taken directly from the Zip file, and are not localized, whereas the “Mapped” values show localized text.  
     
   For example, the mappings for the Basic life cycle template example to be used for replicated WTPart objects are:
4. Repeat this process for all object types being replicated.
5. When all mappings are complete select “OK” or “Apply” to save them.
6. After an import is complete you can verify that the mappings are correct by viewing the Life Cycle History of replicated objects. Note that if the information page of a replicated object does not already display the Life Cycle History, a new tab can be added to include this information. For example, the replicated Pump object described in 2.4 shows the following Life Cycle History:  
     
   