Upgrading to Percussion CM System 7.1.0

This document serves as a guide on how to upgrade to CM System 7.1.0. Upgrading your CM System to 7.1.0 is currently supported from Version 6.7 or 7.0.3.

The next few sections discuss all the necessary steps required to upgrade your CM System from Version 6.7 or 7.0.3 to 7.1.0.

Upgrading from CM System 6.7 to 7.1.0

The following steps outline upgrading CM System from 6.7 to 7.1.0.

1. Shut down CM System and take a backup of the CM System tree (/Rhythmyx) and the entire CM System database. It is very important to take a backup of your CM System server before applying any patches or upgrades

2. Download and install the latest CM System 6.7 Patch from ftp.percussion.com. This patch will be located within /Patches/6.7/

3. If Percussion's Professional Services installed its PSOToolkit on your CM System, you want to update this by downloading and installing PSOToolkit 6.7 from ftp.percussion.com. This PSOToolkit will be located within /Releases/6.7/PSOToolkit/

    a. If you have customized this PSOToolkit, you will need to make all the necessary updates to your custom version of the PSOToolkit to use the new annotation class introduced in this release. Refer to 'Appendix: Updating Relationship Effects' section within this document. This step is necessary only if you have customized the installed PSOToolkit. Note that you will not be able to upgrade to CM System 7.1.0 unless you have made these changes to your custom version of PSOToolkit.

4. If *Segmentation* or *Landing Page* Solutions were deployed on your CM System implementation, update these by downloading and copying the following jars from ftp.percussion.com into its 'lib' directory. These jars will be located on the Solutions ftp site within /7.1.0/Solutions/

    a. soln.segment - /Rhythmyx/AppServer/server/rx/deploy/rxapp.ear/rxapp.war/WEB-INF/lib/soln-segment-rx.jar

    b. soln.landingpage - /Rhythmyx/AppServer/server/rx/deploy/rxapp.ear/rxapp.war/WEB-INF/lib/soln-landingpage-rx.jar

5. Download and install CM System 7.1.0 from ftp.percussion.com. This installer will be located within /Releases/7.1.0/

6. Read the following documents once CM System is successfully upgraded to 7.1.0

    a. CM System 7.1.0 Release Notes

    b. CM System 7.1.0 Release Notes Guide

Upgrading from CM System 7.0.3 to 7.1.0

The following steps outline upgrading CM System from 7.0.3 to 7.1.0.

1. Shut down CM System and take a backup of the CM System tree (/Rhythmyx) and the entire CM System database. It is very important to take a backup of your CM System server before applying any patches or upgrades

2. Download and install the latest CM System 7.0.3 Patch from ftp.percussion.com. This patch will be located within /Patches/7.0.3/

3. If Percussion's Professional Services installed its PSOToolkit on your CM System, you want to update this by downloading and installing PSOToolkit 7.x from ftp.percussion.com. This PSOToolkit will be located within /Releases/7.0.3/PSOToolkit/

    a. If you have customized this PSOToolkit, you will need to make all the necessary updates to your custom version of the PSOToolkit to use the new annotation class introduced in this release. Refer to 'Appendix: Updating Relationship Effects' section within this document. This step is necessary only if you have customized the installed PSOToolkit. Note that you will not be able to upgrade to CM System 7.1.0 unless you have made these changes to your custom version of PSOToolkit.

4. If *Segmentation* or *Landing Page* Solutions were deployed on your CM System implementation, update these by downloading and copying the following jars from ftp.percussion.com into its 'lib' directory. These jars will be located on the Solutions ftp site within /7.1.0/Solutions/

    a. soln.segment - /Rhythmyx/AppServer/server/rx/deploy/rxapp.ear/rxapp.war/WEB-INF/lib/soln-segment-rx.jar

    b. soln.landingpage - /Rhythmyx/AppServer/server/rx/deploy/rxapp.ear/rxapp.war/WEB-INF/lib/soln-landingpage-rx.jar

5. Download and install CM System 7.1.0 from ftp.percussion.com. This installer will be located within /Releases/7.1.0/

6. Read the following documents once CM System is successfully upgraded to 7.1.0

    a. CM System 7.1.0 Release Notes

    b. CM System 7.1.0 Release Notes Guide

Appendix: Updating Relationship Effects

Relationship effects get called in several different ways.

1) When a relationship gets created, destroyed, or updated
2) When an item that is the owner or dependent of a relationship is checked in, checked out, or transitioned through workflow
3) Before an item is cloned, e.g. through copy as new or translate

Effects can be very powerful and can make changes to the system because of these events, or prevent the event from happening. Effects can also cause many problems if not updated carefully. This is usually due to the fact that effects often do not fire in isolation. A whole folder structure could be cloned which would cause many new relationships to be created. Another example is an effect on a new item in a folder could create or modify a different item, which itself would cause an effect to be run when it is inserted into a folder. If an effect takes a long time to run, then it could slow down all of these processes when the impact is multiplied out.

When an effect is called it is passed an IPSExcecutionContext object that can be used to identify where it is being called from. If we assign an effect to the Folder Relationship and an item is added to a folder the effect is fired with the PRE_CONSTRUCTION context that can be tested with excontext.isPreConstruction. This is how the out of the box NavFolderEffect creates a new Navon when a folder is created. We can also check for PRE_DESTRUCTION which would occur when an item is removed from a folder.

The Problem with Move

There is a context PRE_UPDATE in the code and users may have previously created effects that look for this context. The problem of the old mechanism is that a move of an item to another folder would not update a relationship, instead the system would remove the item from the original folder then add the item to the new folder. To the effect this would look like two separate operations which are indistinguishable from a user removing an item from a folder and adding as link to another. Also this meant when the item is being added to the destination it made it difficult if that code needed to know what the original folder was.

This mechanism unfortunately means that we could not prevent a user from removing an item from the last folder it is in and therefore orphaning it as this would also prevent a move. This also would mean that the item may successfully be removed from a folder, but then an error could prevent the item being added to the destination folder, leaving the item orphaned from any folder.

Updated Effect Processing

A move of an item in a folder just modifies the existing relationship changing the owner, and fires any folder effect with PRE_UPDATE context.
As with all the pre effect contexts the relationship is passed to the effect. This is the state of the relationship. Before it is persisted to the database, therefore an error will prevent the change.

e.g. excontext.getCurrentRelationship()

For an update we also would like to know what the current state of the relationship is in the database. We can use this to get the current owner of the relationship if it is being changed in a move.

e.g. excontext.getSourceRelationship()

If a set of items are being processed, e.g. moving a few items into a new folder then, all the effects fire before any of the relationships are modifying, therefore it will not leave the relationships in a half changed state.

A move uses a database update rather than removing and adding a new relationship leaving no chance that a failure will leave the item in no folder at all.

Because PRE_UPDATE is being called for a folder move, code that was attempting to track a move with a remove and will not be called at for a move event unless PRE_UPDATE is being handled. This could cause unknown behavior and an inconsistent system. All effect code needs to be checked to see if it will handle this PRE_UPDATE correctly.

To confirm that the code will work we have designed an annotation that must be applied to the effect classes to tell the system which contexts it is applicable for. This has the added benefit that it allows for auto configuration of the settings when the effect is applied to a relationship.

There are several changes that may be required, and some that are optional.

Clone or Translation Effects
Your code is handling PRE_CONSTRUCTION only for a clone or translation relationship
This may be the case to run some behavior when a item is initially translated or cloned. In this case you will not need to make change to your code, just add the annotation. Clone relationships and translation relationships are never currently modified

Clone and Translation effects normally do not handle PRE_DESTRUCTION or PRE_UPDATE as these relationships are only removed or modified on purge. If you have other code that is specifically removing or updating these relationships you would need to take this into account with any custom clone or translation effects you also have.

Folder Relationship Effects

Your code is handling PRE_CONSTRUCTION only for folder relationship
This is probably because you want to do something when an item is added to a folder. Previously this would have also been called when an item is moved into a new folder, this may or may not have been expected behavior. Now this will not get called on the move. If you do need your code to work in the same way as before you can add PRE_UPDATE and handle it in the same way as PRE_CONSTRUCTION, or you may choose that you only want to handle when an item is newly added to a folder or copy as link.

Your code is handling PRE_CONSTRUCTION and PRE_DESTRUCTION for folder relationship
This is usually because you want to do some action when an Item is added to the folder and clean up when it is removed. The simple change is to allow PRE_UPDATE and handle it in the same way as PRE_CONSTRUCTION currently. Handling PRE_UPDATE specially will probably give you greater flexibility and stability as you can test for consistency and errors in a move differently that add or remove. Handling PRE_UPDATE now allows the NavFolderEffect to also just update the relationship between navon and its new parent when a folder is moved. Before this had to be done in two steps, the navon is removed from the submenu slot of its old parent then added to the new navon submenu slot.

Your code is handling PRE_DESTRUCTION only for folder relationship
Your code is probably cleaning up something when an item is removed from a folder. You may or may not need to do the same cleanup when the item is moved. If so you should add and handle the PRE_UPDATE context

Active Assembly Relationship Effects

There is no set standard to whether AA relationships are modified by modifying the actual relationship, or whether the whole relationship is recreated, this is especially true when APIs are used like the batch importer. PRE_UPDATE should be handled the same as PRE_CONSTRUCTION or by internally calling the PRE_DESTRUCTION followed by PRE_CONSTRUCTION.

PRE_WORKFLOW,POST_WORKFLOW,PRE_CHECKIN,POST_CHECKOUT,PRE_CLONE

The effect processing changes to not affect any effects handling these contexts. The class should still be annotated to say which of these is being handled.

These effects are not directly related to a relationship, but instead are related to an action on an item. When the action occurs on the item all relationships are looked at, whether the item is a dependent of that relationship or owner to see if there is an effect assigned to the type of the relationship. These effects only fire therefore if it is related to a particular relationship. A common thing might be to tie the effect to the folder relationship. That way the effect would fire for any item that is in a folder. If the item is in two folders , e.g. due to copy as link. The effect would get called twice, once for each relationship.

These types of effects also have a direction or endpoint. This specifies if the effect should fire on relationships where the current item is the owner, or dependent, or we are interested in both sets of relationships. The new annotation allows us to specify this in the code, or allow the user to specify