Module 2: XML

As you may know, there are a lot of XML files in the X Rebirth game itself, ranging from production specification (i.e., how to make E-Cells), station definitions, galaxy layouts, to AI scripts, MD scripts and other unexpected things.

Basically everything in X Rebirth that is not part of the "core" system is XML.

XML Patching Guide

For further details, there is a page talking about patching XMLs, and it can be found here. This page offers an overview of what can be done.

Note that when you apply replace or remove patches, ***make sure you match your file structure and file location with the target XML you wish to replace or remove***, or else funny things might occur.

For your interest, starting from v4.0 RC3, you can add silent="true" attribute to any add, replace or remove node to suppress errors when any is produced during loading. This can be useful to create overrides to other mods as 'optional' content and not show up as errors when the other mods are not present.

Add Patch

This patch method adds new content into the vanilla XML when the game processes the mods.

NOTE: You can directly type the XML nodes when adding instead of a diff-add node set, like in the language files, instead of doing <diff><add sel="/languages"><page id=""/></add></diff> you can just directly do <languages><page id=""/></languages>

Replace Patch

This patch method replaces contents in existing (other mods count as "existing" too) XML with something you specify.

NOTE: The content you replace with should match the type of the original content, so after replacing, there will not be any part in the XML "standing out from the rest" because it has been replaced wrongly.

Remove Patch

This patch method removes content in existing (other mods count as "existing" too) XML. To clarify, this method clears the node itself, not replace with white text.

Good Practices

Be Specific

This is simple to understand. Consider the following example, patching the file "/libraries/wares.xml":

```
<diff>
   <replace sel="/wares/ware/production/@amount">50</replace>
   <replace sel="/wares/ware/production/@time">600</replace>
   <replace sel="/wares/ware/production/@amount">50</replace>
   <replace sel="/wares/ware/production/@time">600</replace>
</diff>
```

The game will throw an error because it does not know what you are talking about, or precisely, which ware and which production you are talking about.
In these situations where there are multiple nodes of the same tag under the same node (common in index files), try to specify like:

```
<diff>
   <replace sel="/wares/ware[@id='fusionreactor']/production[@method='default']/@amount">50</replace>
   <replace sel="/wares/ware[@id='fusionreactor']/production[@method='default']/@time">600</replace>
   <replace sel="/wares/ware[@id='fusionreactor']/production[@method='omicron']/@amount">50</replace>
   <replace sel="/wares/ware[@id='fusionreactor']/production[@method='omicron']/@time">600</replace>
</diff>
```

Here, I used the format [@attribute='value'] (notice the single quotation marks) to specify the ware fusionreactor (Fusion Reactors) and its production method default and omicron (Default and Omicron Lyrae respectively). This patch file replaces the values such that "a group of 50 Fusion Reactors will be produced in an interval of 600 seconds in an hour" instead of the existing values.

Be Considerate/Be Specific 2.0

This situation is a bit rare, but might occur when you do not know how to "be specific" when replacing something. Here is an example:

You saw an Energy Array in an Albion zone, and feel very sad because it is not a complete build of the Energy Array. You would like to mod such that it is a complete build of the Energy Array, and you type something like (**This is a pseudo-code; it shows you the idea instead of the actual code**):

<code>

Replace Concealed Hideout (AES Energy Array 1, AES Energy Array 2, AES Energy Array 3, AES Energy Array 4, AES Cell Fab Matrix, Build Spot 1, Build Spot 2), Far Out (...), Albion (...)

with Concealed Hideout (AES Energy Array 1, AES Energy Array 2, *AES Energy Array 3 (...)*, AES Energy Array 4, AES Cell Fab Matrix, Build Spot 1, Build Spot 2)

</code>

You make your mod, and happily load it into the game. You are now satisfied that the AES Energy Array 3 ins finally complete.

This may seem like a happily ever after, but what if you load another mod into the game, and it has the following code:

<code>

Replace Concealed Hideout (AES Energy Array 1, AES Energy Array 2, AES Energy Array 3, AES Energy Array 4, AES Cell Fab Matrix, Build Spot 1, Build Spot 2), Far Out (...), Albion (...)

with Concealed Hideout (*PMC* Energy Array 1, *PMC* Energy Array 2, *PMC* Energy Array 3, *PMC* Energy Array 4, *PMC* Cell Fab Matrix, Build Spot 1, Build Spot 2)

</code>

Since both mods are replacing the same zone, **funny results will occur**. For some users, their end result will be "AES Energy Array 3 is a complete build", and for the others, theirs will be "All NPC stations in Concealed Hideout, Far Out, Albion are owned by PMC", **depending on the load order (which you, as the modder, do not have much control on)**. No errors will be thrown since technically, there is no error - all XML syntax checks would be OK.

To solve this problem, each of the mods can be rewritten as:

<code>

Replace Build Plan (...), AES Energy Array 3 (...), Concealed Hideout (...), Far Out (...), Albion (...)

with *Build Plan*

</code>

<code>

Replace Owner, AES Energy Array 1 (...), Concealed Hideout (...), Far Out (...), Albion (...)

with *PMC*

</code>

In this way, when both mods are loaded, the end results will always be "AES Energy Array 3 is a complete build" AND "All NPC stations in Concealed Hideout, Far Out, Albion are owned by PMC", reducing potential conflicts, confusions and loss of profitssssss.

Please, be considerate.