Modding Tutorial

v 2.0

How to create a mod

1) Go to C:\Steam\SteamApps\common\Endless Space and copy the Public folder except: Audio, Movies, Registry.xml.
2) Go to My Documents\Endless Space (this is also where your saves are stored).
3) Create a new folder and name it “Modding”.
4) Paste the Public folder inside that new folder.
5) Rename it to whatever you want to call your mod.
6) Create a Zip file and upload it on the internet.

How to install a mod

Assuming you downloaded a zip file containing the desired mod.

1) If not already existing, create a new folder called Modding in My Documents\Endless Space
2) Unzip the file’s content to My Documents\Endless Space\Modding

How to run a mod

First, install the mod (refer to “How to install a mod”). Then:

1) Go to Mods on the title screen.
2) Select your mod.
3) Load it!

How to add a faction trait

1) Open MyMod\Simulation \FactionTrait.xml.

2) Copy the empty trait template at the beginning of the file.

In the XML file

- *Name*: the trait's "id", serves as a tag to identify it in any xml file. Different levels for a same trait should have different names (eg: Crusaders1, Crusaders2, etc.).
- *Root*: If your trait has several levels of improvement, this is the “base” name (eg: Crusaders).
- *Family*: The trait’s category in the GUI’s left column, where the trait will be stored (eg: TraitPopulation).
- *Descriptors*: Use the trait’s *Name*.
- *Prerequisites*: Lists all the traits that must or must not be selected for this trait to be used (eg: **!**AntiCrusaders1,**!**AntiCrusaders2, PreCrusaders, BabyCrusaders).
- *Title*: If you want your faction trait to be localized, use”%” to reference a localization key (eg: %CrusaderTitle) and go to a). Otherwise skip to b).

a) Localized text version: Open Localization_Locales.xml in MyMod\Localization\english (or French /German). Create a new LocalizationPair with the %name (eg: %CrusaderTitle) and the actual name (eg: Crusader Crazy). Translate appropriately in the other Localization_Locales files.

b) Simply write the trait’s name

- *Description*: Same as title.

Your trait is now set up.

3) Open MyMod\Simulation\EmpireTraitDescriptor.xml

4) Copy an entire *SimulationObjectDescriptor* sample, at the beginning of the file.

In the XML file

- *Name*: Refers to the same name as in the FactionTrait.xml file.
- *Type*: The "base" name for this trait (eg: Crusaders).
- *Serializable*: True.
- Inside *SimulationObjectPropertyModifierDescriptors* (plural) is where you will add your trait's modifications. You can add as many *SimulationObjectPropertyModifierDescriptor* (singular) as you want. Here is how it works:
    - *TargetProperty*: The variable you want to modify. The target properties accessible can be found at the beginning of the corresponding XML descriptor files (eg: You want to access the FIDS for your Empire, the property will be in EmpireDescriptor.xml).
    - *Value*: By how much you want to modify the variable (positive or negative).
    - *OperationType*: Percent, Addition or Multiplication.
    - By using *SimulationObjectPropertyBinaryModifierDescriptor* (notice "binary" in the name), you can use two operations at once. Eg:
        - TargetProperty = "Approval" OperationType="Addition" BinaryOperationType="Multiplication" Left="8" Right="4" **=> Approval+ (4*8).** The BinaryOperation is computed with Left and Right, then that result is computed with the (normal) Operation on the TargetProperty.
    - *Path:* Very important. It shows what TargetProperty you want to modify exactly.
        - Eg: You want to modify Approval for all Empires, affecting all their systems then Path="ClassEmpire/ClassStarSystem".
        - However, if you want to modify Approval only for a specific affinity, add that affinity using a ",". Path = "ClassEmpire**,AffinityAmoeba**/ClassStarSystem".
        - Other eg: You want to change the DamageMax for Lasers only. This is the Path you would use: ClassEmpire/ClassFleet/WeaponModule, Laser. It is equivalent to ClassEmpire/ClassFleet/Laser.

5) Launch the game, go to Custom Faction creation, and click on the category that was filed under "Family" in FactionTrait.xml

6) Witness a glorious display of Endless beauty and wonder. Your trait, all fresh and ready to rumble space.

How to add a technology improvement

The Tech tree is separated in 4 branches, each independent. There is therefore an XML file for each branch: TechnologyExpansion.xml, TechnologyWarfare.xml, TechnologyDiplomacy.xml and TechnologySciences.xml.

Let's add a new technology in the Expansion branch. *It is the same process for each branch.*

I) Setting up the Tech's icon in the tree

1) Open MyMod\Simulation **TechnologyExpansion.xml** and copy/paste a <Technology> block.
2) The techs are sorted by "circle". The ones at the beginning concern the innermost circles whereas the ones at the end are the most distant from the center.
3) Set the cost to what you want (industry), isUnique=true.
4) X and Y represent **polar coordinates.** Imagine that the whole tree is based on a cross (North, East, South, West = Warfare, Science, Expansion, Diplomacy). There is one "cross segment" per branch. Each (x,y) is based on their respective segment.
    - **X is the polar angle** (clockwise) from that segment to the tech's icon in the tree. For example in the Expansion tree, a positive value for X will place the tech on the left side. A negative value will place it on the right.
    - **Y is the arc's length**, from the origin to the tech's icon.
5) Category = The tree's category (here: ExplorationExpansion).
6) *PathsPrerequisites* represent faction affinities or traits required to access the tech. The syntax uses classic logic (or,and,!). Eg: $(TraitX) and !$(TraitY)
7) *TechnologiesPrerequisites* is unused (at present).
8) *TechnologiesDependencies* is the field that will create the visual links in the tree. Only list the techs that are "immediate parents" of your tech (or else, witness a super web of interconnected techs).
9) *Unlockable* lists the improvements/changes/modifications that your tech unlocks. Use your unlock's name (ID) and a *Standard* access. That ID references the unlock's description (in another XML file). So make sure you use the same name here as where you describe the unlock! For now, simply copy/paste an existing unlockable, we will create ours in a minute.
10) *GUI* has a title and a description. For a localized version of your text, refer to "How to localize your texts". To add a custom icon, refer to "How to add custom tech icons".

II) Creating the unlockable improvement

1) For the purpose of this tutorial, we will create an empire improvement but it works for any type of improvement. Therefore, open **EmpireImprovement.xml**.
2) Copy/Paste an <EmpireImprovement> block.
3) *Descriptors* is the unlockable tech's ID that is used across the XML files. Make sure you use the same name everywhere. Let's create "ForceField1".
4) *GUI* contains the unlock's description and title. As usual, refer to "How to localize your texts" for modding in several languages. The tech's actual effect(s) will be automatically displayed in III).

III) Applying the actual improvement/changes/modifications that your tech unlocks

1) Before creating the improvement, make sure that your icon is correctly created in the tech tree. Launch your mod and go check.
2) Your icon shows up at the right place, with the right unlock(s) great! Now time to create the unlock's effects. Continuing on our Force Field improvement, open **EmpireDescriptor.xml**. **The descriptors contain the game elements' behavior & stats.** That is where you will access

and modify most variables. All of them are defined in a *SimulationObjectDescriptor* (for a Hero, an Empire, a StarSystem etc.) containing various *SimulationObjectPropertyDescriptor.* Most of them are at the beginning of the corresponding descriptor files.

3) Add a *SimulationObjectPropertyModifierDescriptors* (plural) block. It will contain *SimulationObjectPropertyModifierDescriptor* (singular) blocks that modify the object's values.
4) Here is how it works:
    a. *TargetProperty*: The variable you want to modify. The target properties accessible can be found at the beginning of the corresponding XML descriptor files (eg: You want to access the FIDS for your Empire, the property will be in EmpireDescriptor.xml).
    b. *Value*: By how much you want to modify the variable (positive or negative).
    c. *OperationType*: Percent, Addition or Multiplication.
    d. By using *SimulationObjectPropertyBinaryModifierDescriptor* (notice "binary" in the name), you can use two operations at once. Eg:
        i. TargetProperty = "Approval" OperationType="Addition" BinaryOperationType="Multiplication" Left="8" Right="4" **=> Approval+ (4*8).** The BinaryOperation is computed with Left and Right, then that result is computed with the (normal) Operation on the TargetProperty.
    e. *Path:* Very important. It tells what TargetProperty you want to modify exactly.
        i. Eg: You want to modify Approval for all Empires, affecting all their systems then Path="ClassEmpire/ClassStarSystem".
        ii. However, if you want to modify Approval only for a specific affinity, add that affinity using a ",". Path = "ClassEmpire**,AffinityAmoeba**/ClassStarSystem".
5) Drop to your knees, your eyes watery and with a sly smile, as your über tech is now free roaming the endless space, unforgiving and merciless. Crush'em.

How to localize your text

You've created a mod, but you want your texts, descriptions, etc in French or German too?

1) Whenever you see text (like titles and descriptions) in your modified XML, instead of directly writing the text in that space, use an id like %YourText.
2) Open YourMod\Localization\Localization_locales.xml, go to the end of the file and add a new *LocalizationPair* entry with %YourText.
3) Enter your text between the tags.

How to create a Hero

Heroes are created in *Hero.xml*, with their name, starting level, classes and attributes. As always, localized text should use the format %YourText. For now, the heroes' visual aspect must exist in the database, so choose an existing portrait for both the large and small icons.

The heroes' base stats can be found in *HeroDescriptor.xml*. It is also where you can create new hero abilities. Just like creating traits or improvements, modifications will be made inside a *SimulationObjectPropertyModifierDescriptors* block, with several

SimulationObjectPropertyModifierDescriptor lines. Refer to point 4) of "How to make a faction trait" to know how to use the tag.

In *HeroAbility.xml*, you can describe the dependencies between abilities. Given a block *HeroAbility,* any ability listed in *AbilitiesDependencies* must be unlocked before using the one you're creating. Set *isActive* to true if you want it to be immediately available. Finally, the GUI must use existing icons for now.

How to create a planet anomaly

There are two important files for creating planet anomalies: *Anomaly.xml* and *PlanetDescriptor.xml.*

Visual Aspect

1) Open *Anomaly.xml.* Your anomaly's visual aspect is defined here.
2) Copy/Paste an *AnomalyEffect* block. Name it PlanetAnomalyXX where XX is the next number available. You will use that name in *PlanetDescriptor.xml.*
3) *PlanetRimPower* represents how powerful the visual effect is. The lower the value, the stronger the effect.
4) *PlanetRimColor* and *AtmosphereColor* are R,G,B, alpha (transparency) values for the effect, from 0 to 255.
5) Use an existing prefab if your anomaly requires one. Place it either in *AnomalyPrefabPlanet* or *AnomalyPrefabDummy.*

Effects

1) Open *PlanetDescriptor.xml*
2) Search the comment "Anomaly list".
3) Copy a *SimulationObjectDescriptor* block, name it with the same ID as in *Anomaly.xml*. Use the type *PlanetAnomaly.*
4) To apply effects, refer to point 4) of "How add a faction trait".

Adding the anomaly

1) Open *GalaxySettings.xml*
2) Search "Anomalies"
3) Add a *PlanetAnomalyPerPlanetType* tag with your newly created anomaly for each PlanetType.
4) Add a weight. NB: The keyword used is "probability" but it really is a weight. If AnomalyA has "Probability=1" and AnomalyB has "Probability=2", then B has twice more chances of appearing.

Adding a description

1) Simply open *Localization_Locales.xml* in MyMod\Localization\english (or French , German)
2) Search "PlanetAnomaly".
3) Copy a "title" line and a "description" line, then change the number accordingly.
4) Add your title and description between the tags.

NB: For now, it is not possible to manually add an anomaly to a specific planet. By doing this, you're adding your anomaly to the "pool" of anomalies available in the game .

How to create a battle card

You can create your own battle cards. For now, they have to use the same visual aspect as a card already present in the database.

A battle card is created in *BattleCard.xml*, its effects are described in *BattleCardDescriptor.xml*.

Creating the battle card

1) Open *BattleCard.xml*
2) The GUI contains the classic <Title>, <Description> and <Icon>tags. Refer to "How to localize your text" and use a pre-existing icon (for now).
3) *Family* is the card's type. You can create your own family at the end of the file (search "families"), with its title, description and the card's color.
4) *Counter* is the type of cards that your card will counter.
5) *Actions* is where you list the descriptors that contain your card's effects. These descriptors are themselves described in *BattleCardDescriptor.xml.* An *Actions* tag has 3 *Pack*, "Regular", "Counter" and "Countered". Right now, a card isn't played when countered, the latter is therefore empty. You're free to add a behavior when countered by the enemy. **The "Counter" pack is added to the effects of the "Regular" pack,** it **doesn't replace it**.
6) The tags you can include in a *Pack* are documented at the beginning of the file, in a comment. Most are very specific and their effect directly described in the *BattleCard.xml* file. However, alterations are still in *BattleCardDescriptor.xml*.

Card Effects

1) Open *BattleCardDescriptor.xml.*
2) The structure is similar to all other types of modifications. Most cards will have a *BattleCardDescriptor* for regular behavior, and a *BattleCardDescriptorCounter* for the added counter bonus. Refer to point 3) of "How to add a faction trait" to know how to proceed.
3) As always, make sure you use the same descriptor names from one file to the other, to reference them correctly.

Activating the card

1) Open *Technology.xml*
2) Add an *Unlockable* tag with your card's name and with a "Standard" access.

How to add a galaxy size

1) Open *GalaxySettings.xml*
2) In *BaseOptions*, simply add a *GalaxySize* line with the appropriate values (*width* represents the distance in parsecs)
3) Beware, a very large galaxy will take several minutes to generate.

How to add a custom faction (in XML)

1) Open *Simulation/Faction.xml*
2) Copy a "Faction" tag block.
3) *Gui* represents what will be displayed. If you want to add a custom Icon, refer to the paragraph below.
4) List your Traits separated by ";" after having chosen an Affinity.

How to add a custom faction portrait

Disclaimer: In order for another player in MP to see your custom faction portrait, he has to use the same mod as you. Otherwise, the image will be empty.

Setting up the custom faction folder

1) Inside *YourModName/Gui*, create a *CustomFactions* folder (convenient name to locate all your custom faction graphics).
2) Place all your images in that folder. Supported extensions are jpg and jpeg. The image of the faction that is displayed next to its description can be of any size (let's call it AlienDescriptionImage.jpg). There are two sizes for the leader's portraits:
    a. Small : 32x32
    b. Large: 128*128
    Let's say you have AlienLeaderSmall.jpg and AlienLeaderLarge.jpg

Linking the elements in the folder to the actual faction

1) Assuming you created a faction in *Faction.xml*
2) In your faction block, go to *Gui.*
3) In *Icon*, set the "Small" path to *CustomFactions/AlienLeaderSmall* (without the file extension!).
4) Set "Large" to *CustomFactions/AlienLeaderLarge*
5) Set "Wide" to *CustomFactions/AlienDescriptionImage*
6) Launch the mod and your new faction should appear with your custom images.

How to add custom tech icons

1) In *Yourmod/Gui*, create a folder with the name that you want, eg: *TechIcons.*
2) Place all your tech images there.
    a. They must be PNG files (for transparency).
    b. 64x64 in size.
3) The name you give a specific icon is the one you used in the *Gui* block when creating the corresponding technology. By specifying *TechIcons/MyIcon* in the *Gui* block, the game will use your custom icon.

How to add hero portraits

1) Follow the same method as the one above. For convenience, add your pictures in a *HeroPortraits* folder for example.

2) The only difference concerns the portraits' sizes.
    a. Large: 128x64
    b. Small: 64x32
    c. Wide: 284x128
3) The pictures' references go in *Hero.xml.*

How to add a new planet

It is **strongly advised** to download the CustomPlanetsAndShips mod available in the Modding Forum, under "Tutorials for Modding". The archive contains all (and only) the files necessary to create a mod with planets and ships.

Let's add PlanetPinky, a joyful pink planet. It is advised to rename your modded files with a custom name "eg: PlanetColonization => CustomPlanetColonization". You will then have to reference them in Index.xml, in order for the game to locate them.

Creating the planet's properties & statistics

1) Copy the entire **Plugins** folder to your mod's folder and modify *GalaxySettings.xml* by adding *PlanetPinky* whenever other planets are described. In this file, you will define the probability of luxury resources on your pink planet, the probability of appearance, of anomalies, temples, etc…It is also where you define home systems and starting planets for the factions, should you want to force a faction to start on your pink planet.
   Note that even though you only modified *GalaxySettings.xml,* you still have to include the whole Plugins folder in your mod.
2) The **Simulation** folder contains the planet's properties (in FIDS for example). You only need the following files:
    a. PlanetColonization.xml
    b. PlanetDescriptor.xml
    c. PlanetSizeModifier.xml (will override the planet sizes, explained below)
    d. TechnologyExpansion.xml (assuming you want to unlock your planet in that tree)
    e. Gui.xml
3) *PlanetDescriptor.xml* contains the planet's properties. You define the FIDS there.
4) *PlanetColonization.xml* is the technology to colonize your pink planet. Add the wanted PathPrerequisites and AIPrerequisites. The icons in the Gui tag are the ones that will appear in the tech tree for ColonizePinky. Use classic sizes for that.
5) *PlanetSizeModifier.xml* is a custom file overriding the population on planets depending on their size. It isn't possible to simply "add" the population value line for just PlanetPinky, all populations have to be referenced. For clarity purposes, it is wiser to therefore add a PlanetSizeModifier file containing the pop per planet size.
6) *TechnologyExpansion.xml* is used to display the tech containing your ColonizePinky.
7) *Gui.xml* is the image displayed in Planet View, you have to add one or the mod won't load.
8) The **Localization** folder contains your xml localization files. You don't need to copy all the existing lines, anything that is present in your custom file will override existing localization keys and use the news ones correctly.

9) To teach the AI how to colonize your Planet Pinky, you will have to add AI Parameters. Use existing files as a base (it's a system of weights for FIDS). You don't need to reference your CustomAIParameters.xml file in Index.xml BUT you strictly have to follow the path YourMod/AI/Parameters/yourfileshere.

Importing your textures

Again, it is strongly advised to download the CustomPlanetsAndShips mod, as it contains "base" files for the textures, with the correct nomenclature and sizes.

In Index.xml, add a Templates tag, you will place custom models here.

```
<Templates>
    <Planet Reference=>
        <Prefab>
        <Folder>
        <HaloColor>
            <r>
            <g>
            <b>
```

Reference is the name/path it'll be known as in game. This should not collide with any known prefab, except if you want to "overwrite" the source (aka change the ships of a faction for example)

Prefab is the source you want to base your mod on. Available prefabs:

- PL_Terran_A
- PL_Arid_A
- PL_Arid_B
- PL_Arctic_A
- PL_Barren_A
- PL_Barren_B
- PL_Desert_A
- PL_Desert_B
- PL_GasHelium_A
- PL_GasHydrogen_A
- PL_GasMethane_A
- PL_Jungle_A
- PL_Lava_A
- PL_Ocean_A

Folder is your modded texture folder. It contains all your textures in JPG. We use a system of **cube map**, where square textures are applied on a spheric model (the planet). It is **necessary to respect**

the case for the nomenclature. Eg: PL_Pinky_A_Bk_DIFF.jpg (also try to use files that aren't too large)

- PL_Pinky_A is the name you used for the Reference in the Planet tag.
- Bk = Back
- Bt = bottom
- Fr = front
- L = Left
- R = right
- Tp = Top
- DIFF = diffuse
- SPEC = specular

HaloColor is the general color you want to put on the planet halo. Values go from 0.0 to 1.0 (RGBA values)

How to add a new ship model

It is **strongly advised** to download the CustomPlanetsAndShips mod available in the Modding Forum, under "Tutorials for Modding". The archive contains all (and only) the files necessary to create a mod with planets and ships.

Let's add ship Anna_V_01. Models are OBJ files. In Index.xml, add a Templates tag, you will place custom models here (if you already created one to add a planet, simply continue the tutorial within that Templates tag).

```
<Templates>

        <Ship Reference=  Size=>
            <Prefab>

            <Model>
```

Reference is the name/path it'll be known as in game. This should not collide with any known prefab, except if you want to "overwrite" the source (aka change the ships of a faction for example).

Size : Small / Medium / Large

Prefab is the source you want to base your mod on. It is used to compute collisions, demis, etc. **As of version 2.0, that prefab is fixed. The reason for this is that the Terran Medium has the most generic launch sites (demis) for projectiles.** In later versions, you will be able to define specifically what type of prefab you want.

Model is your modded model. **Use OBJ for the model and MTL for your material.** Base yourself on the "demo" ship contained in CustomPlanetsAndShips (Anna_V_01) for sizes and scale, and make sure your ship has the same orientation as Anna's.

- Small = Anna's size

- Medium = 2x Small
- Large = 2x Medium