Mods Manual

Version 1.0 – 2017 June

Summary

Introduction

Welcome to Endless Space 2!

This tutorial provides an initial, very basic set of instructions to help you to mod the game. More detailed information will be added in the upcoming months.

If you have further questions, please visit our forums:

https://www.games2gether.com/endless-space-2/forums/75-modding

Notice

Achievements blocked

By installing a mod you will automatically prevent the game from completing Steam achievements. This is done because mods can change the gameplay balance and difficulty, making many achievements too easy to obtain.

Repairing the base game

The base version of Endless Space 2 (latest updated version with no mods active) may become unstable after deactivating a mod that you were using. Should this occur, you will have to verify the game files using the following process:

- Right-click on the game in your Steam library, then click on “Properties”
- Go to the "Local Files" tab
- Click on the option “Verify integrity of game files”
- Any missing or corrupted files will be repaired or downloaded
- The repaired game should function normally

Where to start?

If you would like to create your own mod but are not sure where to start, the best place to begin is by sharing ideas and information with the community on our modding forums. Modding tutorials will be added in the weeks after the game launch by both the developers and the community. In addition, some community members may have good ideas but lack the time or skills to do the mod themselves. You might even end up working together with other modders to create more ambitious projects!

Forum Modding G2G : https://www.games2gether.com/endless-space-2/forums/75-modding
Forum Steam Workshop : http://steamcommunity.com/workshop/discussions/?appid=392110

The 3 types of mods

There are three types of mods that are possible:

Standalone: This type of mod should be used for large-scale mods with a lot of different content additions and changes. They are completely independent and cannot be launched with other mods. They replace the Steam “Public” folder, which means everything in the public folder should be present in your mod along with your modifications.

Conversion: This type of mod implies a major change to existing files. Only 1 conversion can be launched at a time, but you can add Extension mods in parallel if you want.

Extension: This type of mod is used for minor changes. You can launch as many as you want, but pay attention to the number next to each Extension as it will determine the order in which it is processed during launch of the game.

Multiplayer games

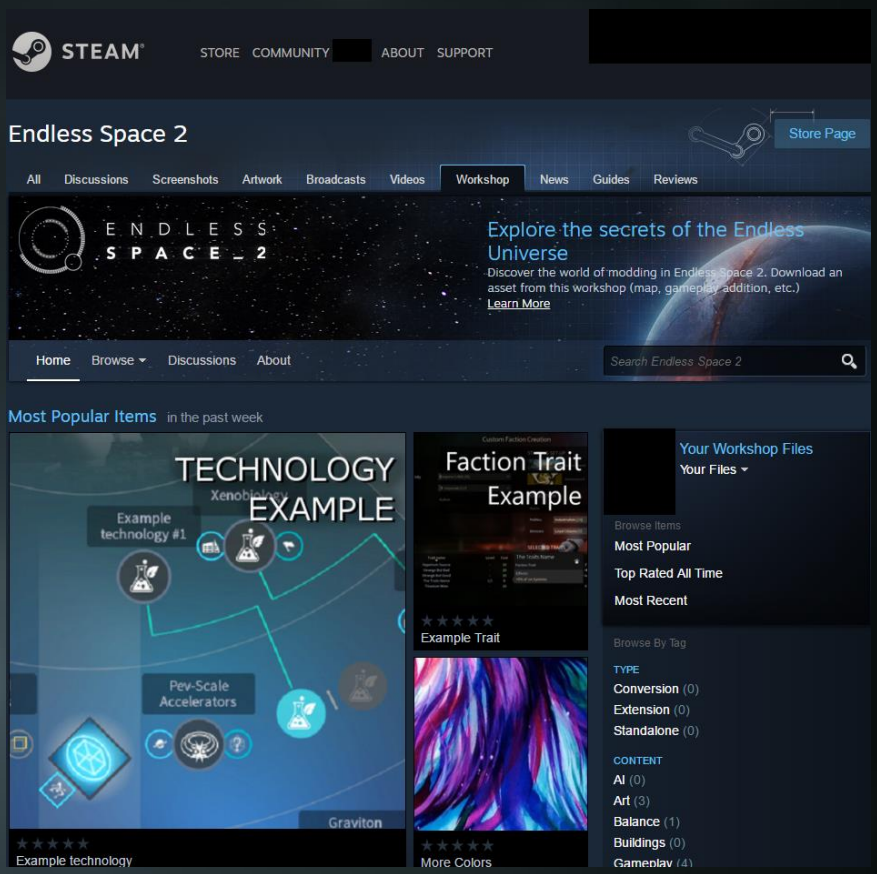
All participants in a multiplayer game must have the same mods installed to be able to take part in the game. In the Multiplayer Lobby you can easily see what mods need to be either installed or deactivated in order to join a particular game.

Steam Workshop

The Steam Workshop for Endless Space 2 has mods created by the community that will allow you to change different aspects of the game—from new gameplay mechanics to changes in the game graphics.

http://steamcommunity.com/app/392110/workshop/

On this page you can see the list of available mods and select the ones you want to try by subscribing to them. Mods you select will be downloaded and ready for use in your next game session. The mods have tags to help categorize or filter them and simplify your searches. For example, if you are looking for mods that will change the skins on ships, you should choose the “Art” tag.



Managing mods

Using Steam Workshop (recommended)

The easiest way to install a mod is to pass via the Steam Workshop, as explained above. Mods you subscribe to will be automatically downloaded when the game starts, and placed in the “Workshop” directory of your Steam installation.

These mods will be updated automatically when a new version is available.

To remove a mod, simply unsubscribe from it in the Steam Workshop.

Manual installation

You have the choice of installing mods yourself, if you prefer.

Create a new directory called “Community” if it does not already exist:

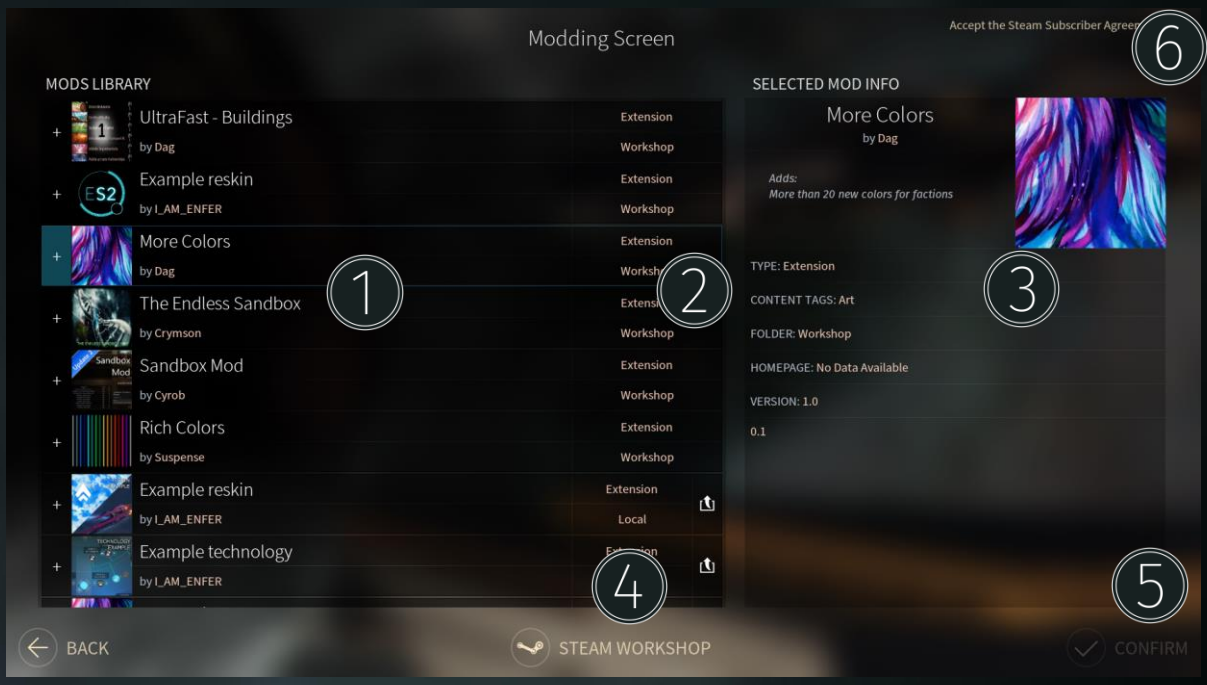
For PC installations, put the directory here: My Documents\Endless Space 2\Community

For Mac installations: go in “Library” then open/create the folder Application Support\Endless Legend\Community

Put the contents of your modding folder into the Community directory to make them visible to the game.

Mod Manager

You can access the Mod Manager from the Game Start screen by clicking on "Mods".



1. The left column is a list of all the mods that are installed and available (all the mods for which you have a Steam Workshop subscription). To activate one or more mods, click on them and then click the "Confirm" button to reload the game with these mods. To remove the mods, unselect them and click "Confirm" again.
2. The button to the right of the mod title is used to upload the mod to the Steam Workshop. If you have not created the mod, the button will not be available.
3. Mouse over a mod to see its description in the right-hand column.
4. The "Steam Workshop" button lets you access the mods directly from the game, using the Steam Overlay.
5. The "Confirm" button allows you to reload the game with the selected mods.
6. The "Accept the Steam Subscriber Agreement" button allows you to open this page directly from the Steam Overlay. Note: It is necessary to accept the Steam Subscriber Agreement in order to upload or update a mod.

How to create a mod

This section contains information that will be helpful for learning how to do basic mods of the game.

What are the limits?

All of the files in your "Public" installation directory can be modified:
C:\Program Files (x86)\Steam\steamapps\common\Endless Space 2\Public

You can play around with them, using their structure and their functions as inspiration to try different things.

If you aren't sure how to proceed or want more information, you can ask for help from the community and the mod developers on our web site:

Forum Modding G2G : https://www.games2gether.com/endless-space-2/forums/75-modding
Forum Steam Workshop : http://steamcommunity.com/workshop/discussions/?appid=392110

Mods folder

Create a new folder called "Community" if it does not already exist:

- For PC installations, put the folder here: My Documents\Endless Space 2\Community
- For Mac installations, go to "Library" then open or create the folder Application Support\Endless Legend\Community

Put the contents of your modding directory into the Community folder to make them visible to the game.

Starter Pack

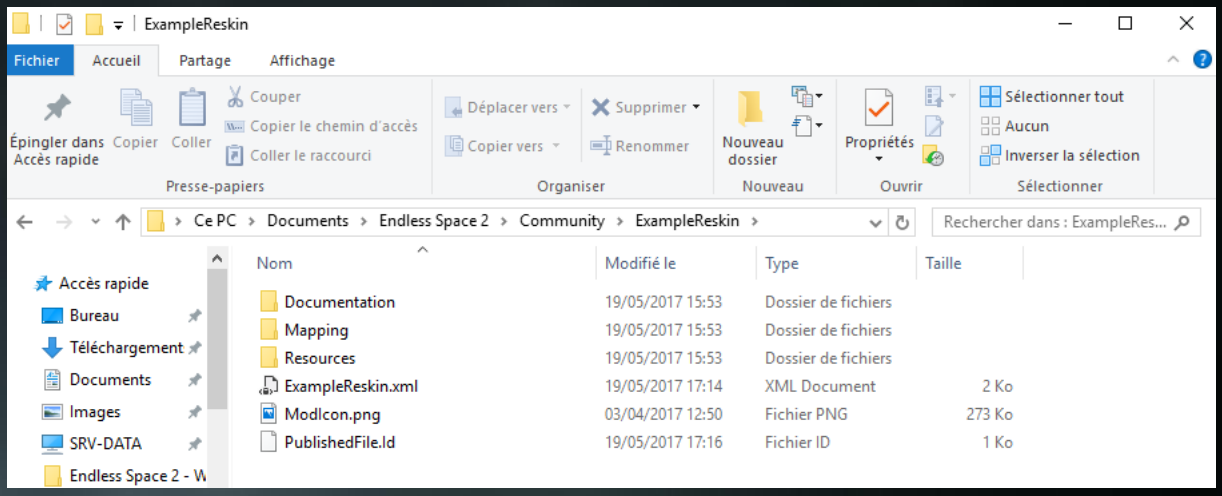
We have put together a starter pack of all the basic elements that you need to start modding. You can download the pack here:

StarterPack.zip

We also have four sample mods that you can download to see the way that the data is organized and to inspire you to find other ideas:

- ExampleReskin.zip
- ExampleTechnology.zip
- ExampleSimpleQuests.zip
- ExampleModule.zip

Structure



Unzip the Starter Pack when you have downloaded it. The pack contains the following elements that will be useful for your Endless Space 2 modding project:

Gabarit.XML

This file contains the tags and file structures that you will need to use to make your mod visible to and usable by the game engine. All of the fields showing "*#########*" will have to be replaced by the appropriate data for your mod (Runtime mod name, Type, Version, etc.). The comments in this file explain what the meaning of these different elements are and provide a sample called "ExampleTechnology".

Documentation Folder

This folder contains various documents that will be helpful in your quest to develop your mod.

DataTypes.txt

This file lists all the different data types that can be edited through XML, as well as their respective files, located in ES2's installation folder's subfolder "Public". It also contains the DatabasePlugin that needs to be called in your XML.

The DataTypes document is regularly updated and new types are added; we will keep this document updated on the forum to be sure that the modding community has all the information they need to mod the features of their choice.

Link to the most recent version of DataTypes : https://www.games2gether.com/endless-space-2/forums/75-modding/threads/25943-manual-for-modding-endless-space-2?page=1

Interpreter.txt

This is a list of the calculations and formulas used in the game.

Simulation.txt

This is information about the game simulation; it will be explained in the next section of this document.

Schemas Folder

This folder contains all of the XML files that you can mod. This should help you understand which value are required, and in what format they need to be, for the mods that you make.

For this to function, the correct file must be called in the header of the XML file that you want to modify. For example, for the "palette.XML" file, you must have this in the header:

```
<Datatable XMLns:xsi="http://www.w3.org/2001/XMLSchema-instance"
XMLns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:noNamespaceSchemaLocation="../Schemas/Palette.xsd">
```

Notes:

- We recommend that you respect this organization of the files and directories; it uses the same structure as the “Public” directory of the game and will help you to easily update and upgrade your mod.
  C:\Program Files (x86)\Steam\steamapps\common\Endless Space 2\Public

- After you upload your first mod to the Steam Workshop, a file called “PublishedFile.Id” will be created and placed in the root of your XML file. Do not delete or modify this as it is needed to make the link with the Steam Workshop and to determine if it is a new mod or an upgrade to an existing mod.

Assets Export

On the game start screen, we have added the ability to export different game assets to facilitate your modding efforts. If you mouse-over the “Mods” button the “Game asset export” button will appear:

Loading all of the available assets in the game can take a long time, and may give the impression that the game has frozen. Please be patient, as there is a lot of content to load!

The following screen will eventually appear, allowing you to export the assets you want to work with:

Selected assets will be exported to the following directory:

Documents\Endless Space 2\Temporary Files

Simulation Basics

Introduction

This introduction to the simulation will attempt to cover as much of the basics as possible without diving too deep in terms of complexity. While this guide will get you started, nothing will ever be as good an example as looking into the files of the game to see how it's done. There are so many cases, exceptions and conditions that it would be impossible to cover everything. So, if you have an idea and can't find what you need in this guide, you should not hesitate to look into the game's XML files to find bits and snippets that you can reuse and modify!

Also, you can find very useful syntax information in the **`Simulation.md`** and **`Interpreter.md`** files (found in *`Public\Documentation`*; you can open them with any text editor such as Notepad++).

Here are a few of the most basic concepts used in the simulation:

- **`Property`**: A Property is kind of like a variable – it's an element that contains a value.

  For example, *`EmpireManpower`* is a *`Property`* and if its *`Value`* is 200 it means you have 200 Manpower in your Empire. A property can be initialized in various ways to fit your needs, using elements like *`MaxValue`*, *`MinValue`*, *`BaseValue`*, *`RoundingFunction`*, etc.

- **`Modifier`**: A *`Modifier`* is an element that operates a modification on the *`Value`* of a *`Property`*. For that it uses a bunch of elements, the most common being *`TargetProperty`* (the *`Property`* you want to modify), *`Operation`* (*`Addition`*, *`Subtraction`*, *`Multiplication`*, *`Percent`*…), *`Value`* (the value of the *`Operation`*) and *`Path`* (the path of your targeted *`Property`*).
- **`BinaryModifier`**: A *`BinaryModifier`* is a basically a *`modifier`* except it does a two-stage operation (such as multiplying the *`Values`* of two *`Properties`* before adding the result to a third *`Property`*), hence its name.
- **`GuiElement`**: A *`GuiElement`* is what you use to properly display information in the GUI such as names, descriptions, icons etc..

Structure

In most XML files you will see text like *`<PathPrerequisite>`* or *`Path=""`*. What follows is a path indicating which element is being targeted or checked.

The structure follows a hierarchy, starting from the Empire and its "child" elements which may have child elements themselves. It can look like this:

`-ClassEmpire`

`----------ClassResearch`

`----------ClassColonizedStarSystem`

`----------ClassColonizedPlanet`

`----------ClassPopulationPlanet`

`----------ClassSenate`

`----------ClassLaw`

So, if you want to reach the population on your planets the path will be:

../ClassEmpire/ ClassColonizedStarSystem / ClassColonizedPlanet/ ClassPopulationPlanet

It's sort of similar to the way you'd browse folders and sub-folders on a computer.

Of course, this path will not be the same depending where you are trying to access the *ClassPopulationPlanet* from, and you might need to navigate "upward" in the hierarchy to go back down on the right path. This is what the periods at the beginning of the path are for:

Let's say you have a hierarchy like *ClassA>ClassB>>ClassC*

"ClassA/ClassB/ClassC" means the check starts at A then ensures B and C are children.
"./ClassC" means that it checks the parent only.
"../ClassB/ClassC" means it will look at where is B, wherever that may be, then see if C is a child.
".../ClassA" = means it will go to the root of the simulation, whatever the location of A is.

You can also check for a specific *SimulationDescriptor* (SimulationDescriptors are often carrying effects, modifiers or serve as tags) by appending it at the end of your path, with a comma – like this:

ClassEmpire/ClassColonizedStarSystem,ColonizedStarSystemStateColony (to check if a colonized star system is a colony (as opposed to an outpost))

What you have to remember is that each time you create an object it has to have a Descriptor to work correctly (a SimulationDescriptor often contains game effects).

The last bit of information to take into consideration is that each class has properties that can also be checked through calculations, rather than by using a path. You can then enter your formulas in *<InterpreterPrerequisite>* instead of *<PathPrerequisite>*

For example, in *ClassEmpire*, you can check several numbers: *AllianceCount*, *WarCount*, *HeroCount*, *EmpirePointStock*, *CurrentTurn*, *EmpireApproval*, etc.

An Interpreter formula would look like this:

```
Property(Context,@'ClassColonizedStarSystem/PopulationParasited', PopulationCount) ge 1
```

Sealed properties

Sometimes you will find a Property that is defined as sealed (by *IsSealed="true"*)

This means the Property can be used but not modified by the simulation, often because they're computed in the code. For example, the maximum movement points of a ship will be defined in a simulation Property but the current movement points of a ship (what it has used and what is left) will be stored and computed in code which mean you won't be able to access it (and you shouldn't try to!).

Basics

Create or modify an effect

The first thing you might want to know as a modder is how to change an effect or how to create a new one.

For that you will need a *SimulationDescriptor*, containing one (or more) *Modifier* or *BinaryModifier* that affects a *Property*.

As an example, let's consider the trait "Businessmen I" that gives +10% Dust on Systems:

```
<SimulationDescriptor Name="FactionTraitBusinessmen1"    Type="FactionTrait">
        <Modifier TargetProperty="SystemMoney" Operation="Percent" Value="0.1"
        Path="ClassEmpire/ClassColonizedStarSystem"/>
</SimulationDescriptor>
```

Let's look at the different elements here one by one:

- **Name**: The name of the *SimulationDescriptor* is pretty much self-explanatory. It's the name you use to refer to it or search for it.
- **Type**: Every *SimulationDescriptor* must have a Type. It is "FactionTrait" in this case.

    - **TargetProperty:** The *Property* you want to modify – basically what you want to change. Here, we're talking about the Dust on a System
    - **Operation**: The type of modification you want to apply on this *Property* – basically how you want to change it. Here, we have a *Percent* but it can be *Addition*, *Subtraction*, *Multiplication*, *Division*, *Percent* or *Power*.
    - **Value**: The *value* of the modification – basically by how much you want to change it. Note that for our example the value is "0.1" due to how *Percent* works. *Percent* increases a value by a percentage that is expressed in on a decimal scale; this means 0.1 is +10%, -0.3 is -30% and 1 is +100%.
    - **Path**: The *Path* is the location of the *TargetProperty* you want to modify in the simulation (this was covered above under the "Structure" heading).
    - **More stuff**: If you don't want your effect to be shown in tooltips, you can add *TooltipHidden="true"* after the path. If you want an operation to take place before or after another you can add something like *Priority="0"* where you replace 0 by another positive or negative number. The higher the number is, the later it will be computed.

For a *BinaryModifier*, see this example:

```
<BinaryModifier TargetProperty="SystemMoney" Operation="Addition"
Left="$(WreckedMothershipCount)"  BinaryOperation="Multiplication" Right="25" />
```

You will notice the *Left* and *Right* parts along with *BinaryOperation*. These are the components, the two parts of the *BinaryModifier*. In a more straightforward form it means adding 25 Dust per Mothership Wreck (Vodyani Ark wreck) on a system.

In further details, the *Left* and *Right* values are computed together using the *BinaryOperation*, then the result is computed with the *TargetProperty* using the *Operation*.

Also, you will note the absence of a *Path*. This is because this *BinaryModifier* can be found in *SimulationDescriptors[ColonizedStarSystem].XML* and it targets a Property *SystemMoney* that is also located in *ClassColonizedStarSystem*: there is thus no need to indicate a path.

Create or modify a GUI element

Modifying elements or creating new ones is one thing, but getting them to display correctly is quite another!

Let's say you create a shiny new trait! However, when you launch the game, all you see is an ugly pink debug text that says *missing GuiElement*…

This is because we have not yet set up anything that tells the game how to render this trait in the GUI. For that you'll need to create a *GuiElement*.

Let's keep the example of the "Buisnessmen I" trait – here's its *GuiElement*:

```
<GuiElement Name="FactionTraitBusinessmen1">
        <Title>%FactionTraitBusinessmen1Title</Title>
        <Description>%FactionTraitBusinessmenDescription</Description>
        <Icons>
            <Icon Size="Small" Path="Bitmaps/Atlased/Headers/HeaderTraits" />
        </Icons>
    </GuiElement>
```

- **Name**: Again, it's self-explanatory. Give your *GuiElement* the exact same name as the element you want it to feedback. The trait is named FactiontraitBusinessmen1 and so is the *GuiElement*.
    - **Title**: This is the name of the localization key associated with the title of your element. It must start with the symbol *%* and could technically be anything, although we strongly recommend to follow the existing convention: *%NameOfYourGuiElementTitle*
    - **Description:** This is the name of the localization key associated with the description of your *GuiElement*. The description is often a more narration-oriented text, usually found in tooltips although there are many different applications all over the game. This is not mandatory; a *GuiElement* can be valid without a description.
    - **Icons:** This is the icon you want to associate with your *GuiElement*. You need to indicate a *Size* (*Small*, *Medium* and *Large* are the most common) and a *Path* so the game knows which icon to display and where it's located.

      Note: The *Path* of an icon has nothing to do with the *Path* we covered in the simulation introduction. For icons, it is the actual path in the game files on your computer. For the moment, you can't import your own icon or picture in the game but we are working on it.

Now you might be wondering how to make your cool new trait display its effects in a tooltip… Worry not, Amplitude has you covered! The tooltip effects are interpreted from the modifiers and automatically generated (as long as the TargetProperty has a GuiElement, usually found in GuiElements[GameVariables].XML)

So now you know how to make a GuiElement… Ready for extended knowledge? Here comes the extended GuiElement!

The ExtendedGuiElement is a GuiElement on steroids that has additional features, one of which can be pretty useful for you modders: The TooltipElement and its EffectOverride.

Basically, this is a lifesaver for cases where an effect is a bit convoluted or exotic and the automated tooltip generation doesn't create proper feedback. For our example, imagine the trait "Businessmen I" had effects that failed to be properly displayed in the tooltip; you would then have to write something like this:

```
<ExtendedGuiElement Name="FactionTraitBusinessmen1">
        <Title>% FactionTraitBusinessmen1Title</Title>
        <Description>%FactionTraitBusinessmen1Description</Description>
        <Icons>
              <Icon Size="Small" Path="Bitmaps/Atlased/Headers/HeaderTraits"/>
              Path="Bitmaps/Dynamic/Improvements/FakeImprovements/EmpireImprovementPlanetLa
              rge" />
        </Icons>
        <TooltipElement>
              <EffectOverride>%FactionTraitBusinessmen1EffectOverride</EffectOverride>
        </TooltipElement>
</ExtendedGuiElement>
```

This will override the tooltip effect, using the content of the provided localization key instead. You will then have to manually write the effect in the localization file. Of course, this shouldn't be the standard way to display an effect since it means that every time you change the effect (say you change it from 10% to 5%) you will have to manually modify it in the localization files.

Thanks for reading this guide, and best of luck with your modding efforts!