The MES mod is a framework that allows us to easily spawn grids in groups with a variety of behaviors, spawn rates, locations, and grid modifications. This guide will introduce the basics of setting up spawn groups and territories for the MES framework and will provide some guidelines for proper documentation and formatting styles.

Working with XML: Formatting Conventions

XML, or eXtensible Markup Language, is as its name tells us, a markup language. As such it is used to organize, or “markup”, data in an easy to process format for both the user and the machine. XML uses an organizational system that divides data into trees, with each data “node” organized as a master node or a “child” of another node. However, the complexities of this language are beyond the scope of this document. As such, to establish a basis for ensuring a consistent format, please follow the below guidelines:

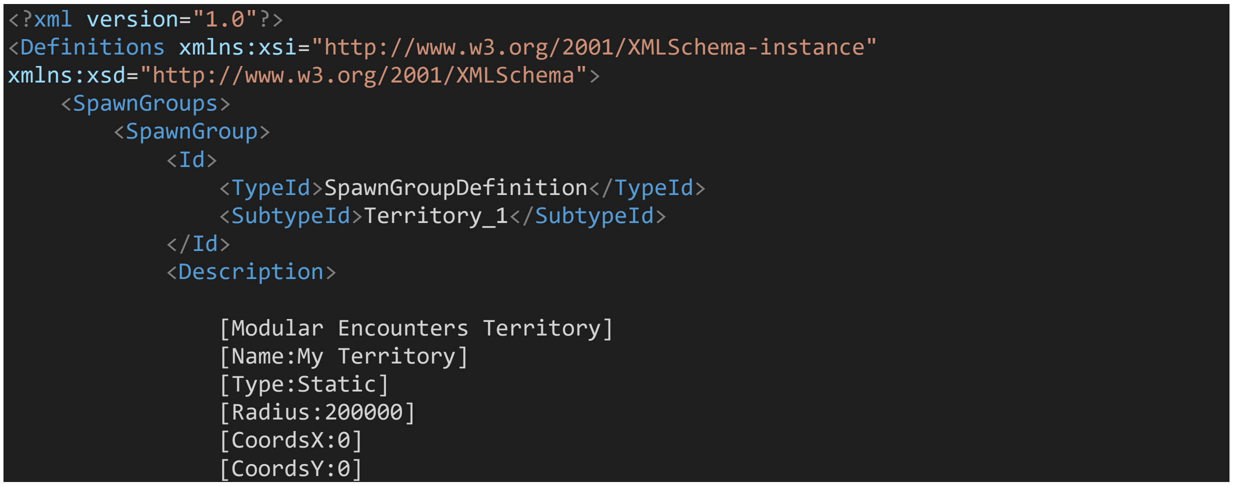
- Always indent child nodes one tab (4 spaces) from the indentation of their parents
- Always indent text contained within a tag that is used for config (eg. the Description tag) one indent past the tag’s indentation
- Use snake-case for file naming (eg. “the best file”.xml becomes “The_Best_File”.xml)

But honestly, just make your XML readable and comment things so things are easier to find. GOOD FILE NAMING IS A MUST!!!!!

Creating Territories

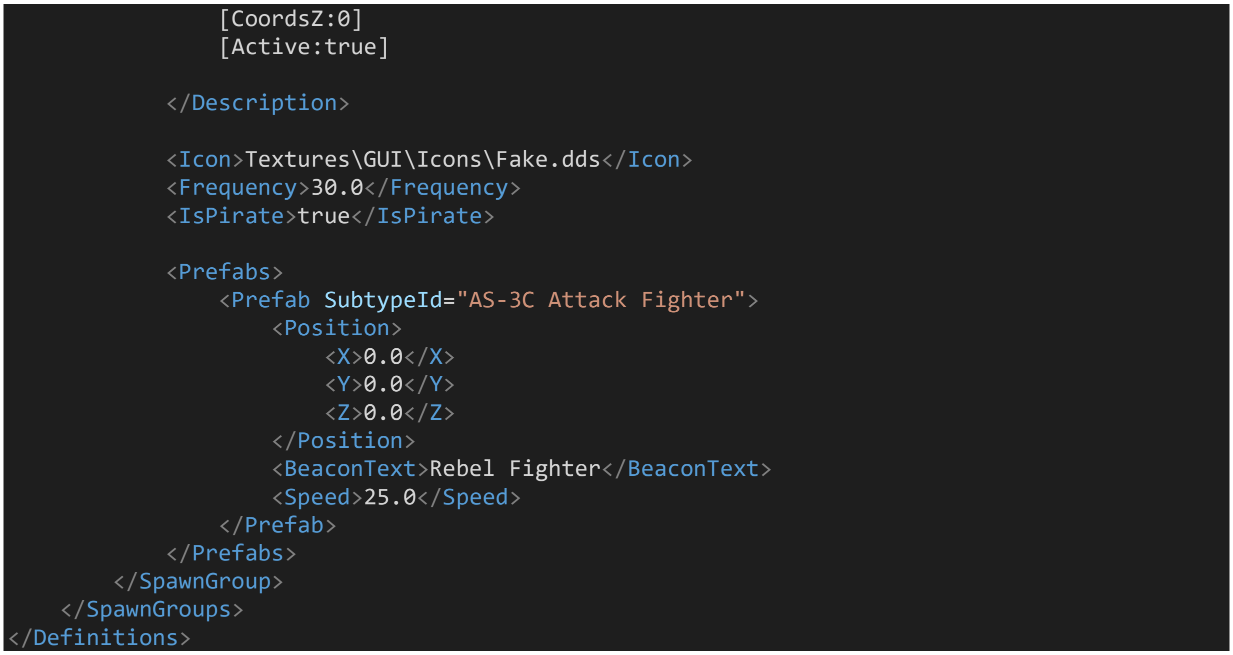
One way the MES framework allows for spawn control is by defining special spheres as territories that allow for selective spawning of grids. In this section we will look at both the Planetary and Static methods for defining territories for the selective spawning of internal spawn groups. All territory definitions must be made in a sbc file in the Data directory.

The sbc file must have an xml prolog, version 1.0 is recommended, and use a “Definition” tag as the root node. Territories are defined using the “SpawnGroup” structure and must contain all tags typically required for a spawn group. Figure 1 depicts a barebones territory



```
<?xml version="1.0"?>
<Definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SpawnGroups>
        <SpawnGroup>
            <Id>
                <TypeId>SpawnGroupDefinition</TypeId>
                <SubtypeId>Territory_1</SubtypeId>
            </Id>
            <Description>

                [Modular Encounters Territory]
                [Name:My Territory]
                [Type:Static]
                [Radius:200000]
                [CoordsX:0]
                [CoordsY:0]
```



```
                [CoordsZ:0]
                [Active:true]

            </Description>

            <Icon>Textures\GUI\Icons\Fake.dds</Icon>
            <Frequency>30.0</Frequency>
            <IsPirate>true</IsPirate>

            <Prefabs>
                <Prefab SubtypeId="AS-3C Attack Fighter">
                    <Position>
                        <X>0.0</X>
                        <Y>0.0</Y>
                        <Z>0.0</Z>
                    </Position>
                    <BeaconText>Rebel Fighter</BeaconText>
                    <Speed>25.0</Speed>
                </Prefab>
            </Prefabs>
        </SpawnGroup>
    </SpawnGroups>
</Definitions>
```

Figure 1: A Basic Implementation of a Territory. This file would be saved as Data\Territories.sbc

There are several important things to note about Figure 1. Firstly, there are two tags that appear to do the same thing: the “SpawnGroups” tag and the “SpawnGroup” tag. This type of organization is also repeated with the “Prefabs” tag and the “Prefab” tag. The reasoning behind this structure is to allow for multiple “SpawnGroup” tags inside the “SpawnGroup” parent and multiple “Prefab” tags inside the “Prefabs” parent. Consequently, if one wanted to add additional territories to this document all one would need to do is add more filled “SpawnGroup” nodes.

Secondly, the description tag contains some information that is stored as plaintext and does not use the typical XML syntax that we are used to seeing. That is because the information stored in the description tag is read separately by the MES framework.

Finally, you also may be wondering why we are using “SpawnGroup” tags for defining territories. Shouldn’t we use a “Territory” tag? Well, apparently it was easier for Meridus to use this tag for territory definitions, and as such, just *deal with it*.

What the Tags are For

- **“SpawnGroup”** tags define a new spawn group. To create a new territory, create a new spawn group
- **“ID”** tags both define the type of definition the spawn group is defining, and the Subtype ID of the definition. These can be filled as properties, or as shown in Figure 1, as children of the ID tag
  - I highly recommend snake case formatting for the SubType ID
- The **“Description”** tag contains definitions for configurables within the MES framework. For a complete guide to these configurables, see Meridus’s Wiki for Spawn Groups.
- The **“Icon”** tag is not used by the MES but is required by KSH. Just use the placeholder for this tag as given in Figure 1 (Textures\GUI\Icons\Fake.dds)

- The **“Frequency”** tag is not used for territory definitions and can be any positive floating point number
- The **“IsPirate”** tag is not used by the MES but needs to be set to true to avoid unexpected behavior
- The **“Prefabs”** parent must contain one valid prefab. Please see the section about creating spawn groups for more information. The prefab is not used but *must* be valid or a loading error will occur when the mod is used.

Setting Up the Description Tag

Static Territories

Static territories are spherical and defined by a single three-dimensional point in space and a radius. The spatial coordinate defines the center of the territory, where the center of the sphere rests, and the radius corresponds to the center of the sphere.