THE ISLE OF ANSUR
BETWEEN SHADOWS & LIGHT

Official modding guide

Legend

D — directory

F — file

Types of mods

⚓ Back to Table of Contents

Isle of Ansur currently has four types of mods (or "packs" which is more correct name):

- statpack
- themepack
- worldpack
- scripts

Every mod can constitute one or more of those pack types, as they all interact with different part of the game. Let me shortly introduce them.

I. Statpack

Statpack is type of pack that builds the backbone of whole game by implementing data related to statistics - a core feature of RPG game!
Statpacks let you add new races, classes, religions, origins, attributes, skills, but will also allow for adding new creatures in the future.

II. Themepack

Themepack is usually a pack structured around game's look - it introduces colour theming, images seen as a background for menus, as well as GUI elements.

III. Worldpack

Worldpack is type of pack that is not yet available in the game, however in future updates it will be core part of worldbuilding of Isle of Ansur.
It will allow you to create locations, NPCs, quests, and basically all the environment game puts you in.

IV. Scripts
Scripts aren't pack type per se, because they usually serve as addition of features to one of previous pack types.
Additionally, their implementation right now is far from good or optimised, thus using them risks high possibility of resulting with buggy behaviour.

How to add mods to the game?

⚓ Back to Table of Contents

Adding mods to the game in *pre-alpha 3* requires manual task of building whole mod structure in game files - initially this version was meant to come with *pre-alpha 2* unpacker, but it was delayed due to heavy rewrite third version got.

In the future, it will be enough to pack your whole mod structure into .zip file and then put into *mods* directory - mods will be unpacked automatically during start of the game, as it was in second IoA release.

More on how to build the mod will be explained in next section.

Creating statpack

⚓ Back to Table of Contents

Statpacks are made in *stats* directory in your IoA folder. Their structure is similar to other packs:

- ❂ Main IoA folder[D]
    - ❂ stats[D]
        - ❂ statpack ID[D]
            - ✵ contents

The ID is very important: it is what let the game distinguish between mods.
One mod can use multiple IDs, however this is rarely useful.
What's more important is that IDs should be unique enough to not be overwritten by another mod.
IDs are currently unchangeable, by which it means that changing the ID will break saved games.

How to write proper ID?

Ideally, IDs should contain only lowercase latin letters and underscores instead of spaces - this ensures it is compatibile everywhere and does not produce buggy behaviour.
However, since there's no guardrails for it yet, game won't put any penalty on not following those rules, except for when you use ":" which is restricted keyword (it can't be put as folder name on Windows, so it shouldn't be even possible to use it for ID).

Statpack structure

While we discussed place where statpack is located, we haven't dived into how statpack structure looks like.

- ❂ Statpack ID^{D}
    - ❂ lang^{D}
        - ✷ english.toml^{F}
    - ❂ classes^{D}
    - ❂ races^{D}
    - ❂ origins^{D}
    - ❂ religions^{D}
    - ✷ attributes.yaml^{F}
    - ✷ skills.yaml^{F}
    - ✷ perks.yaml^{F}
    - ✷ info.toml^{F}

Two main parts of statpack is language file - located in *lang* folder - and statpack information file - being *info.toml* file. All other elements are entirely optional.
We will cover first one, since for now *info.toml* isn't used. This will change in next version.

Language file

Language files are TOML files that allow you to write translations of your mod elements to English (required) and other languages. They use the same naming as official languages used by IoA, meaning that there are only those languages available at the moment:

- English - being default and required option
- Polish - being indev language, not fully translated even in vanilla

Language file is structured around *keys*, which are semi-IDs made by you in mod features that guide them into specific text in language file.
Let me showcase an example.

```
custom_key = "Translated text"
      (A↑)              (B ↑)

       (A↓)             (B ↓)
another_key = '''
Multiline text, which can use
More lines than one and be formatted <b>with HTML</b>
(making newline uses <br> automatically, though!)
'''
```

In this example, we can see two elements: key (A) and content (B). Just remember about this system, because it will be clearer how this works once we create new element for IoA.

Speaking of which...

Creating Features

⚓ Back to Table of Contents

Statpack elements are made with use of two systems: JSON files put into directories, and YAML files put in main directory.

I. Race

⚓ Back to Table of Contents

Creating race is made using JSON files. You do it simply by creating *races* directory, and file of your choice with .json extension.

- ❂ Statpack ID^D
    - ❂ lang^D
        - ✷ english.toml^F
    - ❂ races^D
        - ✷ example_race.json^F
    - ✷ info.toml^F

Simply adding the file isn't enough and may result in even crashes, but that's how you initialise new races in general.
Remember that race filename is used to create unique race ID, consisting of [statpackID]:[race filename].

Once we created our JSON file, we should fill it with contents. The only thing you need to include as requirement is *key* field, but there are many fields you may want to use as well.

Field	Usage
key	Translation ID that points to respective key in language file
attributes	Attributes that this race has bonus for during character creation
power	Modifiers to four powers IoA bases some of statistics
skills	Skills this race has bonus for during character creation
magic	List of magical skill bonuses given to specific magic types during character creation
names	Lore-friendly names that will be suggested during character creation, separated by genders
perks	Perks that your race may use as their special ability

Feel free to refer to this cheatsheet above to implement fields and understand their purpose.

Below is example race file to understand how fields should look like and how you should write your JSON file.

```
{
    "key": "example_race_key" (IA)
    "attributes": {            (IB)
        "strength": 2,
        "agility":   2
    }
}
```

IA. Translation key
While translation key system is pretty straightforward - you put your key (in our case, *example_race_key*) in race file, and then head to translation file and writes its translation:

```
example_race_key = "My Best Race Ever Created"
```

There's one small caveat.
You need also one more key being used in translation file, which is made out of your translation key and *_descr* being added.
This is description key and it serves purpose of describing your race to the player.
While it can be one-lined, it is common to use multilined string for it, so it is easier to format the description without using HTML <br> symbol too much.

Example translation file for our example can look like that then:

```
example_race_key         = "My Best Race Ever Created"
example_race_key_descr = '''
This race is very good at everything, but not perfect in anything!
It was created as God's creation like Frankenstein, made out of
pieces taken away from best mortals: Da Vinci, Einstein, Picasso
and thousands others!
Unfortunately somehow God made a mistake and while watching
Loki series, he put some tricksteryto that race.
'''
```

IB. Attributes

Attributes are those big parts of player's character. They are used as a basis for many actions in game, and also decide heavily on skill improvements.
They can't be raised after character creation except for when your character levels up - so each attribute point pushes character in certain direction.

Attributes are made as a dictionary - those knowing code will already knowing what it means, but for those new to this concept, I'd say that this is exactly the system we are using already - so that

pairing of *key* and *value*, like in *"key": "example_race_key"*.

There are only six attributes in vanilla IoA, but more can be added by mods. Balanced race build needs to sum attributes to 4 points (you can use negative values to balance).
Here is cheatsheet of attributes available in vanilla:

Attribute	Description	ID
strength	Strength affects melee fight effectivity, inventory load and other factors relying on muscle power.	ansur:strength
agility	Agility is about speed of various actions, so it tells how quick your character is performing actions.	ansur:agility
endurance	Endurance is art of resilience in situation where character has some physical tension affecting them.	ansur:endurance
charisma	Charisma is all about communication and personal impression on others.	ansur:charisma
perception	Perception tells how good character is in realising certain elements of environment around them.	ansur:perception
intelligence	Intelligence is huge factor on how easily character learns and how good at magic they are.	ansur:intelligence

What's important to mention is that you can use both name of attribute, as well as attribute ID for vanilla attributes - meaning that *strength* is equal to *ansur:strength*.
This is not the case with modded attributes, where ID needs to be explicitly written.

IC.