Events Design Handbook

June 10, 2022

Contents

1 Introduction

Modding allows you to make your own content for Shadows of Forbidden Gods, and upload it for all to enjoy. Please note that sadly this process is both difficult and technical support cannot reliably be provided. While you may ask questions regarding mod implementation, this is considered outside the scope of the game's normal technical support, so your questions may go unanswered. It is simply not possible for design time to be allocated to debugging third party mods.

Currently, modding takes the form of custom events. Events are the core method by which lore is expressed to the player in Shadows of Forbidden Gods, and is designed to as easily moddable as possible by players. Each event consists

of an image, a text description, and between one and four responses for the player to take.

Events can be presented to the player in a number of instances, the primary ones of these being:

1. Mid-Challenge, every 10 turns an agent spends on a challenge
2. On movement, when an agent arrives in a location (good for keeping the player aware of which agent is involved and where they are)
3. Per-turn, potentially selecting a location at random
4. On exploration of ruins
5. Interpersonal, with two people involved

2 Event Folders

Events are stored on disk in separate folders, in the 'data' folder in the Shadows of Forbidden Gods main installation folder (For example C:/Program Files (x86)/Steam/steamapps/common/Shadows of Forbidden Gods/data).

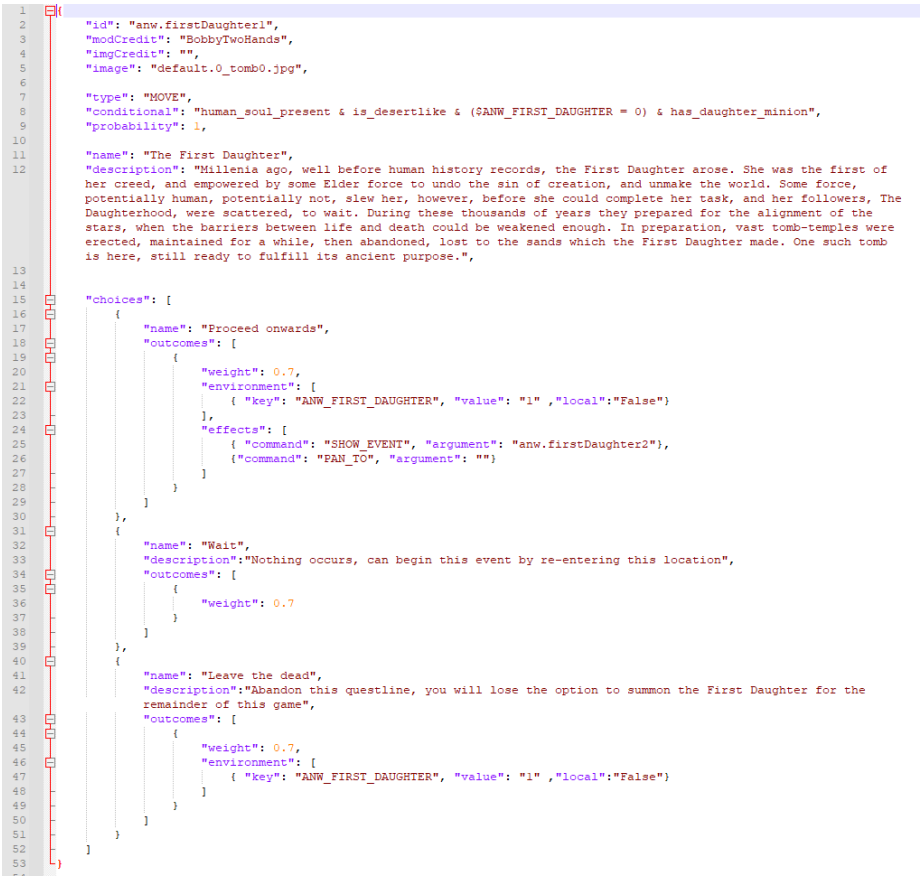
This will already contain a set of folders in events, which can be visited to see the implementation of the existing events, and added to to test mods before releasing them to the public. At the time of writing, there are three mod folders, 'default', for core events, 'eng' for a secondary repository of mid-challenge events, and 'rm' for "removed". Moving a folder into RM will hide it, as only top-level folders are read from, which may be useful to you in testing your events.

Making a new folder is the best start to adding new events, allowing you to form a set of events and images, which are automatically loaded into the main game, allowing you to test them in action. In this folder you'll want to put all your JSON files, one for each event, and whichever images you'll require (although you can refer to events and images from other mods or core events). Notepad++ is a free and lightweight editor, which I personally recommend for the task of event creation.

3 Anatomy of an Event

Events have a set of elements. Most but not all of these are required to function. It is recommended you copy existing events as far as possible, and adapt them to suit your purposes, as opposed to creating them entirely from a blank document.

ID The Id is the unique identifier the game will use to address this event. No two events may share an ID, regardless of if they share the same directory. As such, the best practice is to employ a prefix, ideally three letters, which precedes every one of your mod's events, followed by a full stop and then your event name.



```
{
    "id": "anw.firstDaughter1",
    "modCredit": "BobbyTwoHands",
    "imgCredit": "",
    "image": "default.0_tomb0.jpg",

    "type": "MOVE",
    "conditional": "human_soul_present & is_desertlike & ($ANW_FIRST_DAUGHTER = 0) & has_daughter_minion",
    "probability": 1,

    "name": "The First Daughter",
    "description": "Millenia ago, well before human history records, the First Daughter arose. She was the first of
    her creed, and empowered by some Elder force to undo the sin of creation, and unmake the world. Some force,
    potentially human, potentially not, slew her, however, before she could complete her task, and her followers, The
    Daughterhood, were scattered, to wait. During these thousands of years they prepared for the alignment of the
    stars, when the barriers between life and death could be weakened enough. In preparation, vast tomb-temples were
    erected, maintained for a while, then abandoned, lost to the sands which the First Daughter made. One such tomb
    is here, still ready to fulfill its ancient purpose.",


    "choices": [
        {
            "name": "Proceed onwards",
            "outcomes": [
                {
                    "weight": 0.7,
                    "environment": [
                        { "key": "ANW_FIRST_DAUGHTER", "value": "1" ,"local":"False"}
                    ],
                    "effects": [
                        { "command": "SHOW_EVENT", "argument": "anw.firstDaughter2"},
                        {"command": "PAN_TO", "argument": ""}
                    ]
                }
            ]
        },
        {
            "name": "Wait",
            "description":"Nothing occurs, can begin this event by re-entering this location",
            "outcomes": [
                {
                    "weight": 0.7
                }
            ]
        },
        {
            "name": "Leave the dead",
            "description":"Abandon this questline, you will lose the option to summon the First Daughter for the
            remainder of this game",
            "outcomes": [
                {
                    "weight": 0.7,
                    "environment": [
                        { "key": "ANW_FIRST_DAUGHTER", "value": "1" ,"local":"False"}
                    ]
                }
            ]
        }
    ]
}
```

Figure 1: default/move_firstDaughter.json

These three letters should be chosen to be fairly unique, to avoid them being used by another mod. By prefixing all your IDs with these three letters, you greatly reduce the probability that another mod will share an event name, even if mods are using fairly generic ID names such as "agent_enters_city".

ModCredit This is what will be displayed along with every event your mod employs. Leave empty ("") to show no credit.

imgCredit If the image requires its own credit, this field can be used to assign a credit or brief caption to the image

image This is the image your event will use. It can employ any image loaded by any mod, as long as it is referenced correctly. The name must be the mod's internal name, followed by a full stop, followed by the file name. Your mod's

internal name may be different from your three letter prefix, and is addressed later. 'default' and 'eng' are the two event groups employed already by the game while 'bonus_insanity' is used by the example mod.

type This indicates when the game will check an event, to determine if it should be displayed, and assign it a 'context'. As of Version 0.5, the event types are: LOCATION, PERSON, UNIT, WORLD, MOVE, MIDCHALLENGE, P2P, MOURNING and INERT. Every event must have one of these types, written in upper case. The way in which an event is invoked determines what is 'accessible'. If a person event occurs, for example, you can test conditions of that person (is_ruler, (gold $>$ 50),is_chosen_one...), and perform actions on that person. The conditions and actions will be discussed later. This section details the elements accessible for each type. Five elements can be accessed within a context: a person, a unit, a location, a second person referred to hereafter as 'other' and the map itself. The map is always accessible, conditions and actions relating to it are always accessible.

1. LOCATION events are checked once per turn for every location. If a location matches the event's condition, it rolls its probability. If its probability is true, the event may fire (maximum of one event per turn). If this is true, an event's context will include a location, and conditions checking properties of a location will check this one. If the location has a human ruler present, that person is accessible for checks. Person checks which receive a null person (locations with no ruler) will always be false.

2. PERSON events are checked once per turn for every person. If the person has a location, such as a ruler, a hero or an agent, the location will be accessible. If the person has a unit, that unit is will be accessible.

3. UNIT events are checked once per turn for every unit (hero, agent or other). The unit will be accessible, along with its location, and person if there is one.

4. WORLD events are checked once per turn. Only the map will be accessible.

5. MOVE events are checked whenever an agent moves (only units under the player's control are affected, heroes do not have on-move events available to them). The unit will be accessible, along with its location, and person if there is one.

6. MIDCHALLENGE events are checked periodically while agents are performing tasks. The accessibility is identical to UNIT, with the agent, the person the agent represents and the location accessible. Note that challenge type conditions exist, but do not require any special access, they are functional if ever a unit is accessible.

7. P2P person-to-person events are checked once per turn, but only on a subset of people. The game identifies a subset of individuals as "cast members" who are interesting to the player. This is done for both gameplay and performance reasons. All agents are cast members, and humans will be added to the cast based on the number of messages featuring them, their combat with agents and their proximity to agents. The Chosen One is always a cast member. Person-to-person events will have the same elements as the PERSON event, giving access to the location and to a unit if there is one, but also give access to the other person, and to a set of conditionals which check their relationship, such as "houses_match" to check if they are from the same House or "at_same_location".

8. MOURNING is a special case P2P event, checked each turn, where the 'other' person will always be dead and mourned by the primary person.

9. INERT events are special, and accessible primarily by being created by other events, allowing multiple events to chain together immediately, based on user decisions. They inherit the accessible context of the event which created them.

To change an event from a mid-challenge to a movement, for example, it is as simple as replacing MIDCHALLENGE with MOVE. Note that the MOVE event would occur every time an agent moves to a new location, so would require a variable to be read and written to, to avoid it happening constantly, due to MOVEs being used a lot more than MIDCHALLENGEs.

Conditional The conditional is the check which governs whenever an event can be shown. It will be discussed in its own section later. Note: FOR IMPLEMENTATION REASONS, AN 'INERT' EVENT MUST ALWAYS HAVE A CONDITIONAL WHICH RESOLVES TO TRUE, SUCH AS (1=1).

Probability If a non-INERT event is checked and found to pass its conditional, the probability is tested. If the game rolls a number between 0 and 1 lower than the probability value, the event can be displayed to the user.

Name The title of the event, displayed at the top

Description The main text of the event. Certain keywords can be replaced, to adjust for gendered language, specific names or interpersonal relationships. Otherwise, the text will be displayed as is. Note that "
n" can be used to insert a linebreak to create paragraphs.

Choices These are the buttons available to the user and the responses they will entail. They will be discussed in depth in their own section.

4 Conditionals

Conditionals are sequences of tests which eventually come together to form a singular true/false value. Various properties, both true/false (referred to hereafter as 'boolean' properties, after their inventor, George Boole), and numeric exist. Boolean properties such as asking of a person "is_chosen_one" returns true if the person is the chosen one, and false if they are not, or if no person is accessible. Numeric values for example include "gold" which returns the amount of gold a person has (if there is one accessible, 0 if there is none). Boolean properties can exist in isolation, so "is_chosen_one" is a valid conditional, and a PERSON type event could use that alone, and would then occasionally trigger with the Chosen One as the target. Numeric values cannot, "gold" by itself is not a valid conditional, but "gold $>$ 25" is. They can be combined through various operators, listed below, to form large sequences of checks which all resolved to form a single true/false boolean.

Parentheses/brackets can be employed, to create structures such as "is_chosen_one & (gold $>$ 25)". The order of operations is undefined, so employing brackets in almost all cases is highly advised, to force the conditional to check stuff in the order you wish it to. Numeric operators can be employed, but only ADD, SUBTRACT and MODULO are implemented, so structures such as "$(gold + hp) > 25$" is possible. MODULO is highly useful in setting up periodic events, with such conditionals as "$(turn\%25) = 3$" triggering once every 25 turns, starting on turn 3.

Boolean operators accept a boolean value on both sides and result in a boolean. For example "(is_chosen_one & is_male)" will be true if the event is testing a person or unit who is the Chosen One and the Chosen One happens to be male. The exception to this is the negation operator "!", which takes only one conditional and returns its inverse, such as "!is_chosen_one" which will be true of any person or unit which is not the Chosen One.

The Boolean Operators available are

1. ! The negation operator. Inverts the truth value of any condition following it. e.g. "!is_male" is true for all female characters

2. & The AND operator, returns true if both sides of the statement are correct, for example "is_female & (gold $>$ 25)" is true if and only if the person or unit is a female character with gold above 25.

3. | The OR operator, returns true if EITHER sides of the statement are correct, for example "is_female | (gold $>$ 25)" can be true for any character who has more than 25 gold, even if they are male, and will be true for any female character, including female characters with more than 25 gold.

4. = The EQUALS operator, returns true if both sides have the same truth value.

Numeric operators take two numeric values, and either turn a boolean or another numeric value. Returning a boolean allows them to act as a conditional,

or to be compared against other booleans, such as "$gold > 25$" being a valid conditional, or allowing it to compare against a boolean as in "$(gold+hp) > 25$". Here the $>$ operator is returning a boolean, while the $+$ is returning a numeric value.

The Numeric Operators available which return a boolean are

1. = The EQUALS operator, returns true if both numbers are equal
2. > The GREATER THAN operator, returns true if the number on the left is larger
3. < The LESS THAN operator, returns true if the number on the right is larger

The Numeric Operators available which return a numeric value are

1. + The ADD operator
2. - The SUBTRACT operator
3. % The MODULO operator

4.1 Variables

As well as reading values direction from the world/person/unit, certain pieces of data can be written and read by the events themselves. Events can write to an 'environment', which can store a 'variable' for future events to read. For example, an event could trigger once by checking if it's written to its variable, discovered it has not, triggering and in so doing writing to that variable, and then not firing again.

This is seen in Figure 1, with the variable $ANW_FIRST_DAUGHTER. Variables are exclusively numeric values, and in this case the conditional, where it checks ($ANW_FIRST_DAUGHTER = 0). This is true by default, as a variable which has not been written to defaults to zero. When the event's first or third option is chosen they write to this variable, setting it to 1. The next time the condition is checked, ($ANW_FIRST_DAUGHTER = 0) returns false, so the event does not trigger.

Variables can be written to either an element accessible to the event, or to the map itself. This particular variable was written to the map, so all events will have access to it. Variables are shared between events, and stored to the saved game.

Conditionals employing variables must indicate they intend to use a variable by prefixing it with '$', as in $ANW_FIRST_DAUGHTER. The variable itself, when written to, is ANW_FIRST_DAUGHTER without the dollar sign. Writing to variables will be covered in the second relating to choices, as they are written to by the choices outcomes.

You cannot inform the system which context you want to read the variable from. $ANW_FIRST_DAUGHTER here is stored on the map, and so all events

will access it, but variables can be written to people, for example. In the demonstration mod "voices in the dark" the variable "$INS_BONUS_INSANITY_CHARACTER" is used, which indicates that this particular person is the one involved in the story. This allows people, locations or units to be tracked between events. When a variable is referenced, the system will automatically check all available options, to see if the variable is present on the world map, the person, the second other person, the unit or the location. It is not defined which will be searched first, and care must be taken to avoid accidentally using the same variable multiple times, as this may cause events to misfire. Again, to avoid accidentally copying variable names from other mods' events, best practice is to use the three letter prefix, in this case ANW.

5 Keyword replacement

To adapt the event's text to the current context, such as a person's name, gender and location, keywords are replaced before the text is displayed to the user. "%PERSON_NAME mines for gold" for example, if the person referred to by the event is called "Urist McDwarf", will be displayed as "Urish McDwarf mines for gold". The keywords are denoted by the percent symbol.

Text in the main description is updated, along with text in the option button labels, option descriptions and the title.

Keywords are:

1. %PERSON_NAME The person's name, assuming a person is present
2. %OTHER_NAME The other person's name, assuming the event is a person-to-person or mourning event (the person being mourned is always second)
3. %UNIT_NAME
4. %LOCATION_NAME
5. %RULER_NAME The name of the ruler of the location, not the person involved in the event, if the location has one. NOTE: This text will not update if no ruler is present, use only when sufficient conditionals have been used to avoid oddities
6. **Any pronoun** %His, %her, %she %himself.... All will be replaced, matching the capitalisation. Both genders can be employed, and will switch appropriately to the character
7. **Any pronoun for the other person** %His2, %her2, %sh2e %himself2.... The other character's pronouns are replaced by this mechanism, by adding a '2' to the pronoun

5.1 Random Text

You can make an event randomise its text somewhat, by using the MULTI tag. By surrounding a group of options between %MULTI[and]%MULTI, split by |, the keyword replacement system will select one of these options at random. For example, %MULTI[a|b|c|d]%MULTI will display a random letter between a and d. Full sentences can be placed into this system, to allow variety in the mod. Multiple separate MULTI blocks are possible in an event, and they will choose their options independently.

6 Choices

An event is able to specify up to 4 different choices, to present to the user. These will have a name, and potentially a description, which gives more information about each one. Each choice then has at least one 'outcome', but can specify any number of outcomes, which are randomly chosen between. Choices can be locked behind conditions, to prevent them from being available to the user if they don't meet a particular requirement (such as needing an agent with 10 gold to have the option to buy an item).

A choice has a set of elements

Name This is the displayed named of the choice. Its text is updated, using the keyword replacement system, so you can have text reading "Heal %PERSON_NAME" which is displayed to the user as "Heal The Supplicant".

Description This is the additional information displayed when the user has their mouse over the choice's button. It also has the keyword replacement system.

Conditional This is the requirement which must be met for this option to be available. It operates the exact same way as the event's main conditional.

6.1 Outcomes

Outcomes are a set of possible results of the user picking this option. If multiple are present, the game can select between them randomly.

Weights Random selection is done by 'roulette selection'. This is done by comparing the weighting of the outcomes, relative to one another. **The sum of the weights does not need to equal 1.0, nor does the weight equal to the probability of the outcome being chosen**. A weighting twice the weight of another outcome will have twice the probability of being chosen, regardless of the numeric values present. A weighting of 0.8 will be chosen twice as often as one of 0.4. This allows easy addition of new outcomes, with the probabilities

automatically re-adjusting without you needing to recompute the probability to maintain a sum of 1.0.

Description The description of the outcome is a message which will pop-up after the outcome has been chosen. It will be displayed to the player after the event has been dismissed, and can let them know what the random outcome was.

Effects These are the actions taken on the game. They can target the unit, location, person or world, depending on which action is chosen. See the section below for a comprehensive list of every action available. Every effect takes the form of a command and an argument. This argument may be used to determine a property of the action, such as "gain gold" having a numeric argument which tells it how much gold to give. Some effects do not vary with their arguments, but, due to technical limitations, the field must still be provided.

Environment These are the changes to the variables. They take the form of a variable name (which is then used in conditionals by adding a '$' to the beginning, the dollar sign should not be used in the name), a numeric value or a condition which resolves to a number (such as 'gold + turn' to add the gold and turn together and store it into the chosen variable), and a target. By default, the variable saves to the map, but you can also make the variable write to a person or location, to say designate a location to be used by future events. See the 'Whispers in the Dark' example mod for usage of a person-variable to identify who will be targetted by future events. If you wish to attach a variable to a location, person, second person referred to by an event, or unit, you may put one of: "LOCATION", "UNIT", "PERSON", "PERSON2" in the 'local' field, and the variable will be stored there.

7 Uploading the mod

Mods should be tested locally, by adding them to the data/events folder in the shadow's main install. This will be found on steam in your steam library, with the default path being "C:
Program Files (x86)
Steam
steamapps
common
Shadows of Forbidden Gods
data
events". You may of course need to adapt this path if you have your game installed in another steam library or have your steam library in another location. You will want to create a new folder in here, test your mod events work along with their images. The game will automatically load the events from here, and you can test and develop in this environment.

When you are done testing, firstly make a backup or two, then move the mod folder from here to "...steamapps
common
Shadows of Forbidden Gods
modUploadFolder". You may need to create the folder 'modUploadFolder'. Delete all other files in this folder before continuing (if you have already uploaded another mod). In here, you will need to create a parent folder for your mod, and within it create a folder named 'content'. Copy all your files into "...common
Shadows of Forbidden Gods
modUploadFolder
bonus_insanity
content" (mod name will obviously not be 'bonus_insanity', that is the example mod name). This content folder should contain your images and json files (it should have their directly here, not inside another folder within 'content'). You can see examples of this in Figures2 and 3. The outer folder is what will be employed by the workshop, and only the content folder from that will be deployed to the user. As such, two descriptor json files are employed, one for workshop data, one for in-game data.

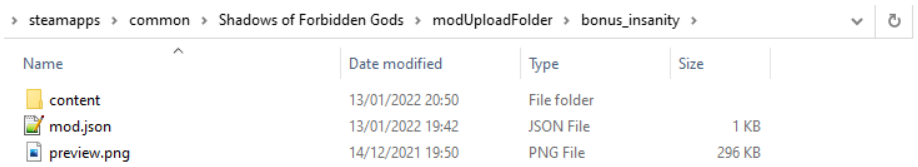


Figure 2: The first part of the upload folder

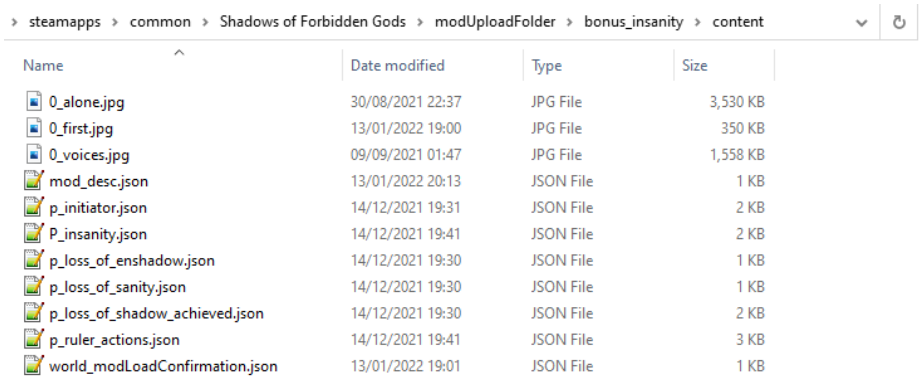


Figure 3: The second part of the upload folder

As before, copying the existing mod is almost certainly the correct choice, as

it has the required files, and simply needs those changed and its content deleted and replaced with your own. Regardless of whether you do this or start from a blank slate, the following files are needed:

preview.png This image will be used to display your mod on the workshop. Ideally a PNG image with dimensions of 512 x 512 pixels.

mod.json This describes the mod to the Steam Workshop, and at time of writing must contain three fields: The title, the Description and the tags. Tags are used to search for mods on the workshop. See Figure4 for example contents. Note the use of square brackets to denote a list, for the tags.

content/mod_desc.json This file describes the mod to the game after it has been downloaded. This is used to ensure version compatibility, and to locate the image files. See Figure 5 for example content. The **prefix** is the prefix used by your images. This must match the one you used in your events, so in this case the event must refer to its first image as "bonus_insanity.0_alone.jpg".

```
{
    "title": "Story: Voices in the Dark",
    "description":"A short story mod about insanity, which will trigger a sequence of events for a ruler if they meet
    the right conditions. Used as a demonstration mod for modders to learn from and repurpose as they see fit, but
    playable in-game",
    "tags":["story mod","demonstration mod"]

}
```

Figure 4: The mod.json file's contents

```
{
    "displayedName": "Voices in the Dark",
    "prefix": "bonus_insanity",
    "versionsSupported":["0.5","0.6"],
    "modCredit": "BobbyTwoHands"
}
```

Figure 5: The mod_desc.json file's contents

One all files are present and appear correct to the game's validation system, a button will appear in the bottom right of your game's main menu, labelled 'upload mod'. Clicking this button will either upload the mod, if you have configured it correctly, or provide more error messages. Once uploaded, you can view your mod on Steam's Workshop.

You can see an example of a correctly configured the example bonus_insanity mod which has all the files ready for uploading, including the correct folder structure.

8 List of Conditions with their types

Map Fields

---

turn : number (integer)

seed : number (integer)

enshadowment : number (integer)

temperature : number (decimal)

panic : number (decimal)

has_enthralled : boolean

power : number (integer)

god_is_snake : boolean

god_is_iastur : boolean

god_is_vinerva : boolean

god_is_ophanim : boolean

god_is_mammon : boolean

victory_percent : number (integer)

Person Fields

---

is_sane : boolean

shadow : number (integer)

gold : number (integer)

is_from_normal_soc : boolean

is_cast_member : boolean

is_agent : boolean

is_chosen_one : boolean

is_mourning : boolean

number_of_friends : number (integer)

is_male : boolean

is_female : boolean

is_ruler : boolean

person_preference_ambition : number (integer)

person_preference_cooperation : number (integer)

person_preference_combat : number (integer)

person_preference_cruel : number (integer)

person_preference_danger : number (integer)

person_preference_deep_ones : number (integer)

person_preference_discord : number (integer)

person_preference_gold : number (integer)

person_preference_madness : number (integer)

person_preference_orc : number (integer)

person_preference_shadow : number (integer)

person_preference_undead : number (integer)

is_hero : boolean

is_deepOne : boolean

has_call_of_the_abyss : boolean

is_birthday : boolean

stat_might : number (integer)

stat_intrigue : number (integer)

stat_command : number (integer)

stat_lore : number (integer)

is_infamous : boolean

person_index : number (integer)

person_house_index : number (integer)

sanity : number (integer)

max_sanity : number (integer)

Person 2 Fields

---

other_is_sane : boolean

other_shadow : number (integer)

other_gold : number (integer)

other_is_from_normal_soc : boolean

other_is_cast_member : boolean

other_number_of_friends : number (integer)

other_is_agent : boolean

other_male : boolean

other_female : boolean

other_ruler : boolean

other_menace : number (integer)

other_profile : number (integer)

other_is_deepOne : boolean

person2_index : number (integer)

person2_house_index : number (integer)

Unit

---

challenge_is_lore : boolean

challenge_is_command : boolean

challenge_is_intrigue : boolean

challenge_is_might : boolean

kills : number (integer)

has_daughter_minion : boolean

menace : number (integer)

profile : number (integer)

minion_one_special_value : number (integer)

minion_two_special_value : number (integer)

minion_three_special_value : number (integer)

is_agent_baroness : boolean

is_agent_courtier : boolean

is_agent_cursed : boolean

is_agent_doctor : boolean

is_agent_harvester : boolean

is_agent_hierophant : boolean

is_agent_monarch : boolean

is_agent_shadow : boolean

is_agent_supplicant : boolean

is_agent_survivor : boolean

is_agent_trickster : boolean

is_agent_warlord : boolean

is_agent_warlock : boolean

is_agent_human : boolean

HP : number (integer)

maxHP : number (integer)

Location

---

is_coastal : boolean

infiltration : number (integer)

is_ruins : boolean

is_human : boolean

is_town : boolean

is_village : boolean

is_hamlet : boolean

is_city : boolean

is_metropole : boolean

is_wonder_font : boolean

is_wonder_deathIsland : boolean

is_wonder_entrance : boolean

is_orc_settlement : boolean

is_orc_fortress : boolean

is_orc_spelltwisters : boolean

is_orc_menagerie : boolean

is_orc_empty_shipyard : boolean

is_orc_shipyard : boolean

is_witch_coven : boolean

is_empty : boolean

is_desertlike : boolean

is_desert : boolean

is_dry : boolean

is_snow : boolean

is_arctic : boolean

is_arid : boolean

is_drycold : boolean

is_grass : boolean

is_highland : boolean

is_jungle : boolean

is_swamp : boolean

is_plains : boolean

is_sea : boolean

is_tundra : boolean

is_volcano : boolean

is_under_fog : boolean

location_shadow : number (integer)

location_index : number (integer)

is_ducal : boolean

is_capital : boolean

has_ancient_ruins : boolean

human_soul_present : boolean

settlement_defences : number (decimal)

settlement_max_defences : number (decimal)

settlement_being_razed : boolean

modifier_level_arcane_fortress : number (integer)

modifier_level_arcane_secret : number (integer)

modifier_level_armoured_populace : number (integer)

modifier_level_banditry : number (integer)

modifier_level_bribed_guards : number (integer)

modifier_level_choking_spores : number (integer)

modifier_level_death : number (integer)

modifier_level_deep_ones : number (integer)

modifier_level_devastation : number (integer)

modifier_level_doubt : number (integer)

modifier_level_faith : number (integer)

modifier_level_fleeting_servant : number (integer)

modifier_level_hunger : number (integer)

modifier_level_geomantic_locus : number (integer)

modifier_level_give : number (integer)

modifier_level_hysterial_tome : number (integer)

modifier_level_laughing_tome : number (integer)

modifier_level_lingering_resentment : number (integer)

modifier_level_madness : number (integer)

modifier_level_malign_catch : number (integer)

modifier_level_misleading_clues : number (integer)

modifier_level_plague : number (integer)

modifier_level_plague_immunity : number (integer)

modifier_level_political_agitation : number (integer)

modifier_level_political_instability : number (integer)

modifier_level_quarantine : number (integer)

modifier_level_take : number (integer)

modifier_level_unrest : number (integer)

modifier_level_vinerva_gift : number (integer)

modifier_level_ward : number (integer)

modifier_level_well_of_shadows : number (integer)

modifier_level_ogre : number (integer)

Two person

---

houses_match : boolean

nations_match : boolean

is_mourning_other : boolean

at_same_location : boolean

is_close_family : boolean

preference_other_to_person : number (integer)

preference_person_to_other : number (integer)

9 List of Actions with their required arguments

ADD_POWER requires: integer argument

GAIN_POWER requires: integer argument

ADD_TEMPORARY_WORLD_PANIC requires: integer argument

Person effects

LOSE_SANITY requires: integer argument

GAIN_SHADOW requires: integer argument

KILL_PERSON requires: verbal

GAIN_GOLD requires: integer argument

GRANT_ITEM_FROM_POOL requires: integer argument

PERSON_LOSE_PREFERENCE_FOR_TAG requires: verbal

PERSON_GAINS_PREFERENCE_FOR_TAG requires: verbal

MOURNING_PROLONGED requires: integer argument

MAKE_CAST_MEMBER requires: verbal

ABDICATE requires: verbal

BECOME_HERO requires: verbal

TEMPORARY_MIGHT requires: Two integers separated by a slash, the first integer being the turns this effect will last, and the second the amount it will change by. Example, for changing the stat by 3 for 5 turns would be "5/3"

TEMPORARY_LORE requires: Two integers separated by a slash, the first integer being the turns this effect will last, and the second the amount it will change by. Example, for changing the stat by 3 for 5 turns would be "5/3"

TEMPORARY_COMMAND requires: Two integers separated by a slash, the first integer being the turns this effect will last, and the second the amount it will change by. Example, for changing the stat by 3 for 5 turns would be "5/3"

TEMPORARY_INTRIGUE requires: Two integers separated by a slash, the first integer being the turns this effect will last, and the second the amount it will change by. Example, for changing the stat by 3 for 5 turns would be "5/3"

SET_AGE requires: integer argument

SET_MIGHT requires: integer argument

SET_LORE requires: integer argument

SET_INTRIGUE requires: integer argument

SET_COMMAND requires: integer argument

MAKE_FEMALE requires: verbal

MAKE_MALE requires: verbal

Person 2 effects

OTHER_LOSE_SANITY requires: integer argument

OTHER_GAIN_SHADOW requires: integer argument

OTHER_GAIN_MENACE requires: integer argument

Location effects

INFILTRATE_POINTS_OF_INTEREST requires: integer argument

UNINFILTRATE_POINTS_OF_INTEREST requires: integer argument

UNINFILTRATE_POINTS_OF_INTEREST_FROM_TOP_POI requires: integer argument

LOCATION_GAIN_SHADOW requires: integer argument

LOCATION_RULER_GAIN_SHADOW requires: integer argument

LOCATION_POP_MULT requires: integer argument

DESTROY_LOCATION requires: verbal

RULER_DIES requires: verbal

EXPLORE_RUINS requires: integer argument

ADD_MODIFIER_DEATH requires: integer argument

CITY_REBELS requires: verbal

PAN_TO requires: verbal

OPHANIM_TAKE_OVER requires: verbal

INCREASE_ALL_DANGER requires: integer argument

SUMMON_FIRST_DAUGHTER requires: verbal

ADD_MODIFIER requires: verbal

REMOVE_MODIFIER requires: verbal

INCREASE_UNREST requires: integer argument

CHANGE_UNREST requires: integer argument

CHANGE_PLAGUE requires: integer argument

CHANGE_PLAGUE_IMMUNITY requires: integer argument

CHANGE_DEVASTATION requires: integer argument

CHANGE_POLITICAL_AGITATION requires: integer argument

CHANGE_POLITICAL_INSTABILITY requires: integer argument

CHANGE_WARD requires: integer argument

CHANGE_WELL_OF_SHADOWS requires: integer argument

CHANGE_DEATH requires: integer argument

CHANGE_HUNGER requires: integer argument

CHANGE_MADNESS requires: integer argument

CHANGE_ARCANE_FORTRESS requires: integer argument

CHANGE_GEOMANCY requires: integer argument

BRIBE_GUARDS requires: integer argument

PLACE_ARCANE_SECRET requires: verbal

SET_UNIT_COMMANDED requires: verbal

CREATE_HERO_WARRIOR requires: verbal

CACHE_GOLD requires: integer argument

Unit effects

GAIN_PROFILE requires: integer argument

GAIN_MENACE requires: integer argument

LOSE_CHALLENGE_PROGRESS requires: integer argument

END_CHALLENGE requires: verbal

ADD_HP requires: integer argument

GAIN_XP requires: integer argument

GENERATE_MINION requires: structured minion description. Format: "ATTACK/HP/DEFENCE/COMMAND COST/NAME/IMAGE WITH PREFIX/SPECIAL_VALUE" Example from First Daughter event chain: "3/2/4/0/Daughter: Thila/default.icon_daughter_sword.png"

The special value at the end is used by other events to detect minions, via conditions which retrieve this special value. It can be left blank or set to 0 without issue. See the 'Astrid' demonstration mod for this in use.

GENERATE_CHALLENGE requires: structured minion description. Format: "Event title/Event description/Team/Profile Gained/Menace Gained/Menace for AI purposes/Complexity/Icon/Outcome event. Example usage: title/desc/-1/4/8/16/32/default.icon.png/anw.outcome

The Team term is one of: -1, 0 or 1. -1 means only 'evil' characters can perform it (the player's agents). 1 means only heroes, and 0 means either The challenge will display the outcome event (anw.postEventTest in the example case). This event will have the completing unit in its context, so you can apply the challenge's outcomes there IMPORTANT: This challenge will be attached to the settlement at the LOCATION the first event occurs at. If there is no settlement, the game will crash. The challenge will disappear if the settlement is destroyed

BEGIN_HIDING requires: verbal

CANCEL_ALL_ATTACKS requires: verbal

RANDOM_TELEPORT_TO_LAND requires: verbal

Battle effects

ATTACKER_ADD_PROFILE requires: integer argument

ATTACKER_ADD_MENACE requires: integer argument

DEFENDER_ADD_PROFILE requires: integer argument

DEFENDER_ADD_MENACE requires: integer argument

DEFENDER_GAINS_TRAIT requires: verbal

ATTACKER_GAINS_TRAIT requires: verbal

INFAMOUS_DEATH requires: verbal

DEFENDER_HATES_ATTACKER requires: verbal

ATTACKER_HATES_DEFENDER requires: verbal

ATTACKER_LOSE_PREFERENCE_FOR_TAG requires: verbal

ATTACKER_GAINS_PREFERENCE_FOR_TAG requires: verbal

DEFENDER_LOSE_PREFERENCE_FOR_TAG requires: verbal

DEFENDER_GAINS_PREFERENCE_FOR_TAG requires: verbal

Two person

DECREASE_P1_TO_P2_RELATION requires: verbal

INCREASE_P1_TO_P2_RELATION requires: verbal

DECREASE_P2_TO_P1_RELATION requires: verbal

INCREASE_P2_TO_P1_RELATION requires: verbal

INCREASE_P1_TO_P2_RELATION_LIMITED requires: integer argument

INCREASE_P2_TO_P1_RELATION_LIMITED requires: integer argument

XENOPHOBIA requires: integer argument

VENDETTA requires: integer argument

Event chain enabler

SHOW_EVENT requires: the id of the event to create

SHOW_EVENT_IF_CONDITIONS_MET requires: the id of the event to create

SHOW_EVENT_IF_CONDITIONS_MET_AND_PROBABILIY requires: the id of the event to create

SHOW_RANDOM_EVENT_WITH_TERM_IN_ID requires: verbal

SHOW_RANDOM_EVENT_WITH_TERM_IN_ID_CHECKING_CONDITIONS requires: verbal

SHOW_RANDOM_EVENT_WITH_TERM_IN_ID_CHECKING_CONDITIONS_AND_PROBABILITY requires: verbal