Warhounds Modding: A Guide for Mod Authors

A mod is a folder of files that add to or change the game's data: texts, ability and weapon parameters, progression, characters, random events, icons.

No coding is required: almost all of the game's data is stored in JSON text files, and a mod contains only what it changes.

This document matches game version 1.0.3.

1. Quick start

- Open the game's main menu, click `MODS`, then `OPEN MODS FOLDER`. This opens the `Mods` folder next to the game executable.
- Create a folder for your mod in it, for example `my_first_mod`.
- Put a `mod.json` file describing the mod in it, along with the files containing your changes (see the sections below).
- Return to the game: the mod appears in the list on its own, disabled. Tick its checkbox and the changes apply immediately, no restart needed.

A minimal mod that renames one ability:

```
my_first_mod/
  mod.json
  Localizations/
    English
```

`mod.json`:

```
{
  "id": "author.my_first_mod",
  "name": "My first mod",
  "version": "1.0.0",
  "author": "Author",
  "description": "Renames Accurate Shot"
}
```

`Localizations/English` (a file without an extension):

```
{
  "ability/title/accurateShot": "Sniper Shot"
}
```

2. Where the game data is

The game's own files are the best reference: open the file you need, find the entry and copy into your mod only the fields you want to change. You don't need to edit the game files themselves.

What	Where
Game configs	`Warhounds_Data/StreamingAssets/Configs`
Game texts	`Warhounds_Data/StreamingAssets/Localizations`
Mods folder	`Mods` next to the game executable
List of enabled mods	`%USERPROFILE%/AppData/LocalLow/EverPlay/Warhounds/ModsSettings.json`
Game log	`%USERPROFILE%/AppData/LocalLow/EverPlay/Warhounds/Player.log`, lines tagged `[Mods]`

3. Mod structure

The folders inside a mod mirror the game's folders. All of them are optional; only `mod.json` is required.

```
my_mod/
  mod.json                      mod description
  Localizations/English         texts (file name = language)
  Configs/Abilities.json        config changes
  Configs/Modules/WeaponModuleConfig.json
  Sprites/Ability/MyIcon.png    images
  Fonts/MyFont.ttf              fonts for new characters
```

The mod.json file

Field	Required	Description
`id`	yes	Unique mod identifier. Don't change it after publishing: the game uses it to recognize the mod in saves and in the list. A convenient form is `author.name`.
`name`	no	Name in the mod list. If not set, the `id` is shown.
`version`	no	Mod version, shown in the list.
`author`	no	Author, shown in the description.
`description`	no	Description shown under the mod list.
`gameVersion`	no	The game version the mod was made for. On a mismatch a warning is written to the log; the mod still

		works.
`dependencies`	no	List of mod `id`s without which this mod won't enable.
`languages`	no	Description of the languages the mod adds to the game (see section 4).

4. Texts and languages

A `Localizations/<Language>` file is flat JSON of the form `"key": "text"`. The file name matches the language name in the game: `English`, `German`, `French`, `Spanish`, `Polish`, `Portuguese`, `Turkish`, `ChineseSimplified`, `Japanese`, `Korean`. The file may also have a `.json` extension.

- **Change a text.** Find the key in the game's file and give it a new value in the mod. All other texts stay as they are.
- **Add a text.** A new key (for example, the name of your ability) is simply added to the file.
- **Translating into only some languages.** If a new key is missing from the mod's file for the current language, the game takes it from the same mod's `Localizations/English`. So always include an English file.

A new language

Put a file named after a language the game doesn't have into the mod, for example `Localizations/Russian`, and describe the language in `mod.json`:

```
"languages": [
  { "language": "Russian", "culture": "ru-RU", "displayName": "Русский" }
]
```

- The language appears in the game settings. Untranslated texts remain in English.
- `culture` sets the date format, `displayName` is the name in the language switcher.
- The language name must be one the engine knows: `Russian`, `Ukrainian`, `Italian`, `Czech`, etc. Arbitrary names are not supported.
- Language flag for the main menu: the image `Sprites/Russian.png`.
- Characters missing from the game's fonts are drawn with system fonts. You can put your own font into the `Fonts` folder (`.ttf` or `.otf`).

5. Configs

A file in the mod's `Configs` folder has the same name as the game file and lies at the same path. It does not replace the game file but is applied on top of it according to these rules:

What is in the mod file	What happens
Object `{ ... }`	Merged with the game object: the listed fields change, the rest stay.
Number, string, `true`/`false`	Replaces the game value.

`null`	Removes the field or entry.
Array `[ ... ]`	Replaces the game array entirely. The exceptions are keyed arrays, see below.
An entry or file the game doesn't have	Added. This is how new abilities, presets, events and scenarios are made.

Change an ability

`Configs/Abilities.json`: Accurate Shot recharges in 3 turns instead of 4:

```
{
  "AccurateShot": {
    "rechargeDuration": 3
  }
}
```

A new ability is an entry with a new name and the full set of fields; the easiest way is to copy a similar ability from the game file. Only an existing ability type (`abilityType`) can be used. The icon is set by the `icon` field and goes into the `Sprites` folder (section 6); the name and description are localization keys.

Keyed arrays

In three configs, array elements are matched by key fields, so a mod lists only its own elements:

Config	Array	Key
`CharactersLevelsData.json`	`LevelsData` of each class	`Level`
`RandomEvents.json`	the whole file	`Id`
`AreaSpawnEnemyPowerConfig.json`	the whole file	`SpawnType` + `PowerMin` + `PowerMax` + `FactionName`

Price of a level 1 Assault; the other levels are untouched:

```
{
  "Assault": {
    "LevelsData": [
      { "Level": 1, "Price": 5000 }
    ]
  }
}
```

To remove an element of such an array, give its key and `"_remove": true`:

```
[
```

```
  { "Id": "Armwrestling", "_remove": true }
]
```

Characters and module presets

A character's parameters (health, speed, accuracy, weapons) are assembled from presets. Presets are in `Configs/Modules/*.json`, character templates in `Configs/CharacterTemplates.json`, and a character refers to presets by name.

A new health preset, `Configs/Modules/HitPointsModuleConfig.json`:

```
{
  "HP_Example": {
    "DefaultHitPoints": 15
  }
}
```

Assign it to a character, `Configs/CharacterTemplates.json`:

```
{
  "shilo8Character": {
    "modules": {
      "HitPoints": "HP_Example"
    }
  }
}
```

- `Weapon` and `Item` in `modules` are arrays: the mod sets the character's whole list of weapons or items.
- A template applies to newly created characters. Mercenaries already hired keep their data in the save and won't change.
- `PlayerDefault.json` (the starting state) applies only to a new game.

Weapons and items

`Configs/Modules/WeaponModuleConfig.json` and `ItemModuleConfig.json`. Example, a rifle's price:

```
{
  "SniperRifleT1": {
    "price": 777,
    "sellPrice": 333
  }
}
```

Difficulty

`Configs/EnemyStatsModifiers.json`: bonuses enemies get from difficulty settings and on hard missions. The five existing values can be changed:

```
{
  "ResilientEnemiesPowerUp": 30
}
```

Random events

`Configs/RandomEvents.json` is an array of events; a mod brings only its own. Individual fields of an existing event can be changed; the lists of participants, conditions and choices (`ParticipantSlots`, `Conditions`, `Options`) are replaced entirely.

Participants, conditions and consequences have a `"$type"` field, the object type. It must come first and match one of the types already used in the game file. The game rejects any other type, and the mod won't enable.

```
[
  {
    "Id": "MyEvent",
    "ImageId": "Armwrestling",
    "TitleKey": "event/MyEvent/title",
    "DescriptionKey": "event/MyEvent/text",
    "IsUnique": false,
    "ParticipantSlots": [],
    "Conditions": [],
    "Options": [
      {
        "ButtonTextKey": "event/MyEvent/choice_1/title",
        "OutcomeTextKey": "event/MyEvent/choice_1/text",
        "OutcomeImageId": "",
        "Consequences": [
          {
            "$type":
"Game.HubSystem.RandomEventsSystem.Consequence.ModifyResourceConsequence, Code",
            "ResourceId": 0,
            "Amount": 1000,
            "ShouldShowLog": true
          }
        ]
      }
    ]
  }
]
```

Enemy spawns

`Configs/AreaSpawnEnemyPowerConfig.json`: which enemy groups appear at which power. An entry is matched by four fields at once; the group list `AreaSpawnPacks` is replaced entirely:

```
[
  {
    "SpawnType": "basic",
    "PowerMin": 0,
    "PowerMax": 9,
    "FactionName": "Daamat",
    "AreaSpawnPacks": [
```

```
      { "AreaSpawnPackID": "big_0_D_1", "Chance": 1.0 }
    ]
  }
]
```

Other configs

Loot (`LootConfig.json`), the shop and mercenary generation (`CharactersShop.json`, `CharacterGeneration.json`), buffs and status effects (`Buffs.json`, `StatusEffects.json`), currency, factions, mission scenarios (`Scenarios/*.json`), dialogues and lines, and clothing sets (`ClothingConfig/*.json`) are changed the same way. A scenario or clothing set with a new file name is added as a new one.

6. Images

- Format: PNG. The file goes at `Sprites/<key>.png`, where the key is the same value given in the config. An icon with `"icon": "Ability/AccurateShot"` is `Sprites/Ability/AccurateShot.png`.
- An image with a key that exists in the game replaces the game's image. With a new key it gives an icon to your ability or item.
- Works for icons of abilities, items and buffs, portraits, and language flags. Images built into interface windows, splash screens and gamepad hints are not replaced.
- Make icons the same size as in the game: large images load with a delay.

7. Mod order and conflicts

- Mods are applied from top to bottom of the list. If two mods change the same value, the one **lower** in the list wins. The order is changed with the arrows in the mod's row.
- When you enable a mod that changes the same values as an already enabled one, the game shows a warning listing the overlaps. `OK` enables the mod, `Cancel` leaves it disabled. The full list of overlaps is written to the log.
- Mods that change different fields of one entry don't conflict: one can change a weapon's price, another its damage.
- If a mod depends on another (`dependencies`), it won't enable without it, and it disables itself when the required mod is disabled.

8. Saves

- The save records the list of enabled mods.
- When loading a save made with a mod that is now disabled or removed, the game warns you. You can cancel loading and enable the mod, or continue at your own risk.
- If you continue without the mod, the game removes abilities, presets and items that no longer exist from character data, and restarts the mission in which the save was made.
- A save made inside a mission whose map or scenario was added by a mod won't load without that mod.

9. If a mod doesn't work

Symptom	Cause and what to do
The mod is not in the list	The mod folder has no `mod.json`, it contains an error, or `id` is not set. The mod folder must be directly inside `Mods`, not in a nested folder.
The mod won't enable, an error window appears	The window names the file and the cause: a JSON error (an extra or missing comma or bracket), a forbidden type in `"$type"`, a missing mod from `dependencies`. The cause stays in the mod description under the list.
`[key/in/brackets]` instead of text	The key is missing both from the game and from the mod for the current language. Add it to `Localizations/English`.
The change didn't apply	Check the file name and path (they must match the game file exactly), the entry name and the fields. Check whether a mod lower in the list changes the same value.
The mod disabled itself	The game couldn't read the config after applying the mod and disabled the mod so as not to be left without data. The cause is in the log.

All mod system messages in the log start with `[Mods]`. Mod files can be edited while the game is running: when you return to the mods window, the game notices the changes and rereads the data.

10. Limitations

- Mods can be enabled, disabled and reordered only in the main menu.
- Mods contain no code. A new ability, scenario step, condition or event consequence is assembled from existing types.
- You can't add a new mercenary class, faction, region or mission type.
- A mod can bring its own models only for clothing and weapons (weapons: section 5.5, clothing: section 12.3 of the config reference). Animations, maps, sound and video can't be changed by mods.
- Images: PNG only, no slicing or atlases.
- Languages: only from the list known to the engine.
- System configs (default settings, settings presets, gamepad layout and the like) can be changed, but they are not supported: changes apply after restarting the game, and the mod author is responsible for the result.

11. Checks before publishing

- The mod has a unique `id`, and `name`, `version` and `description` are filled in.
- The mod enables without an error window, and the log has no `[Mods]` lines with errors.
- Every new text key has an English version.

- A new game starts, a mission can be completed, and a save with the mod loads.
- The mod has been tested together with other popular mods: no unexpected overlaps.

12. Publishing

How to upload a finished mod to the Steam Workshop so other players can find it and subscribe. Screenshots show the English version of the editor; the Russian one differs only in labels.

What you need

- A purchased copy of the game and Steam running on the account you publish from. Limited Steam accounts (less than $5 spent) can't publish.
- Mod Editor and the game folder (the editor asks for it on first launch).
- The mod, open in the editor.

Preparing the mod

- Fill in the mod description: menu Mod → Mod Description (mod.json)…: name, version, author, description. The Tags field takes one or two comma-separated tags from the Workshop list: Weapons, Clothes, Characters, Balance, Localization, Gameplay. Steam rejects a tag that is not on the list.
- Put a preview.png image in the root of the mod folder; it becomes the item's cover in the Workshop (square, at least 512×512, preferably without small text).
- Save the mod (Ctrl+S) and run the validation (F5). Errors block publishing; warnings are up to you.

First publication

- Menu Mod → Publish to Steam Workshop…
- The editor determines which game the mod is open for (Warhounds or Sparta 2035, by the game folder). If unsure, it asks; the item goes to that game's Workshop.
- In the publishing window, check the fields:
    - Workshop title: up to 128 characters, shown in the Workshop;
    - Description: the item description;
    - Tags: from mod.json, can be adjusted here;
    - Visibility: «Hidden (draft, visible only to you)» for a draft; you can make it public later on the item page.
- Click Publish. The entire contents of the mod folder are uploaded, so don't keep extra working files in it.
- On your very first publication, Steam asks you to accept the Workshop terms: an agreement page opens; accept it and click Publish in the editor again.
- Done: the editor offers to open the item page, and workshopId appears in mod.json; from now on the editor always updates this item.

Item page: https://steamcommunity.com/sharedfiles/filedetails/?id=<workshopId>. Players subscribe to it; Steam downloads the mod into the game folder, and it appears in the mods window along with local mods. A published mod gets a link icon to its Workshop page in the

mods window.

Updating

Open the mod in the editor, make changes, save, and again choose Mod → Publish to Steam Workshop…. Now the window offers Update and a Change notes field, a short description of what changed (shown on the item page in the change notes tab). The mod updates automatically for all subscribers.

If something went wrong

Message	Cause
Could not connect to Steam	Steam is not running or you are not signed in
no rights to publish (the game is not owned…)	The game is not owned on this account
limited Steam account	Limited account: spend $5 in Steam
Steam rejected a parameter (…a tag…)	The tag is not on the Workshop list; check the Tags field
Workshop is temporarily unavailable	Try again later

You can delete or hide the item on its Workshop page (Settings, for the item owner).

Warhounds Config Reference

A reference for mod authors: which game configs can be changed, what fields they have and what they mean. Game version: 1.0.3. It covers the gameplay configs: combat, characters, equipment, economy, missions, enemies, random events. The second half (sections 10–14) covers scenarios, combat maps, appearance, dialogues and news. System settings (graphics, controls, interface) are not covered.

How a mod is organized as a whole (folder, `mod.json`, localization, images, the mods window) is described in the document «Warhounds Modding Documentation».

Notation. Type `text key`: the field holds a localization key, and the text itself is set in `Localizations/<Language>`. Type `sprite key`: the image can be replaced with a `Sprites/<key>.png` file. The ⚠ sign in a description means the field's purpose was inferred from its name and examples and has not been confirmed by the developer: check it in the game before relying on it.

1. How a mod changes configs

1.1. Where to put files

The game configs are in the `StreamingAssets/Configs` folder inside the game's data folder. A

mod puts a file with the same name and the same relative path into its own `Configs` folder:

Game config	File in the mod
`Configs/Abilities.json`	`<mod>/Configs/Abilities.json`
`Configs/Modules/WeaponModuleConfig.json`	`<mod>/Configs/Modules/WeaponModuleConfig.json`
`Configs/Missions/BlackMarketConfig.json`	`<mod>/Configs/Missions/BlackMarketConfig.json`

The mod file contains only what changes. It is a patch, not a copy of the config: everything not in the patch stays as in the game.

1.2. Merge rules

What is in the patch	What happens
Object `{ ... }`	Merged with the game object by key, recursively
Number, string, `true`/`false`	Replaces the game value
Array `[ ... ]`	Replaces the game array entirely (exceptions in 1.3)
`null`	Removes the key from the config
A key the game doesn't have	Added: this is how new abilities, weapons and presets are created

Example: change the recharge of one ability and touch nothing else.

```
{
  "AccurateShot": {
    "rechargeDuration": 3
  }
}
```

If several mods are enabled, the patches are applied in the order of the list in the mods window: a mod lower in the list overrides the values of a mod above it. The game warns about overlaps when a mod is enabled.

1.3. Arrays merged by key

In three places an array is not replaced entirely but merged element by element. An element is matched by key fields: if found, the fields are merged; if not, the element is appended to the end.

Config	Array	Key fields

`CharactersLevelsData`	`LevelsData` inside a class	`Level`
`RandomEvents`	file root	`Id`
`AreaSpawnEnemyPowerConfig`	file root	`SpawnType` + `PowerMin` + `PowerMax` + `FactionName`

To remove an element of such an array, give its key fields and `"_remove": true`:

```
[
  { "Id": "Armwrestling", "_remove": true }
]
```

Arrays inside a merged element (`LevelAbilities`, `Options`, `AreaSpawnPacks`, etc.) are replaced entirely.

1.4. Restrictions and safeguards

- The service field `"$type"` is allowed only in random events and only for condition, selector and consequence types (section 9). Any other `"$type"`, as well as `"$ref"`, `"$id"` and `"$values"`, disables that mod's patches with an error in the log.
- If a patched config can't be read, the game uses its own version of the config and the mod stops being applied. The cause is written to the log tagged `[Mods]`.
- The game checks references: ability → status effects, class levels → abilities. A reference to a non-existent identifier is logged.
- Mods are enabled only in the main menu. Many values are copied into characters and items when they are created, so changes are reliably visible in a new game rather than in an old save.
- Not moddable: `DLCSliderData`, `GPURanking`, `Debug/PlayerAvailableItemsDebug`, `BuildLocationsConfig`.

1.5. When changes take effect

Configs	When applied
`Abilities`, `Buffs`, `StatusEffects`, `CharacterTemplates`, `CharactersLevelsData`, `CharacterGeneration`, `CharactersShop`, `InventoryConfig`, `LootConfig`, `Currency`, `Factions`, `EnemyStatsModifiers`, `RandomEvents`, `RandomEventsSettings`, `Missions/BlackMarketConfig`, all `Modules/*`	Immediately when the mod is enabled, no restart
`Paths`, `CharacterRedactorItemsDatabase`, `DefaultSettings`, `SettingsPresets`, `GamepadServiceConfig`	Only after restarting the game
Others	Depends on when the config is

	read. If the log shows the line `Configs changed by mods were loaded earlier and are not reloaded now` with a list of configs, those configs need a game restart

1.6. General conventions

- A value from a list (type, class, faction) is written as a string with its name: `"weaponClass": "Shotgun"`. The allowed names are listed in the appendix.
- An entry's identifier is its object key: `"AccurateShot": { ... }`. Other configs refer to the entry by it.
- Write field names exactly as in the game file, with the same letter case: `CharacterAbilities` and `characterAbilities` are different keys for a patch. Open the entry in the game config before editing it.
- Chances and percentages are given differently in different configs: `0.5` in some, `50` in others. Check the format in the field's table.
- Distances are measured in map cells, combat time in turns, hub time in days.

2. Combat

2.1. Abilities

File `Configs/Abilities.json`. Object: the key is the ability identifier, the value is its description. The game has 333 abilities. Class levels, character templates, weapons, items and buffs refer to the identifier.

A new ability is a new entry with an existing `abilityType`: the type defines the behavior, and a mod can't add new logic.

Main fields

Field	Type	Meaning
`title`	text key	Name
`description`	text key	Description
`icon`	sprite key	Icon, for example `Ability/AccurateShot`
`abilityType`	list	Ability logic. The list of values is in the documentation appendix
`abilityTargetType`	list	What it targets: `Self`, `Sparta`, `Enemy`, `Allies`, `Ballistic`, `AOE`, `AreaAroundTarget`, `Car`, `Mine`

`isPassive`	yes/no	Passive ability: works on its own, has no button
`actionPointPrice`	number	Cost in action points (0–3 in the game)
`isSpendAllActionPoints`	yes/no	Using it burns all remaining action points
`canSpendAttackPoints`	yes/no	Can be paid for with attack points instead of action points
`canBeUsedWhenOutOfActionPoints`	yes/no	Available at zero action points
`canRecharge`	yes/no	The ability has a cooldown
`rechargeDuration`	number	Cooldown in turns
`charges`	number	Number of charges
`limited`	yes/no	The ability is limited by charges: unavailable at zero charges
`freeUsesForMission`	number	Free uses per mission
`freeUsesForTurn`	number	Free uses per turn
`statusEffects`	string array	Status effects from `StatusEffects` that the ability applies
`abilitySettings`	object	Ability parameters: damage, radius, buffs and so on
`triggerOverwatch`	yes/no	Using it triggers enemy overwatch fire
`availableWhileCarryingBody`	yes/no	Available while the soldier carries a body
`abilityWeaponClass`	list	Weapon class the ability is tied to (appendix A.4)
`finishIncapacitated`	yes/no	Only for the `Taser` type: a critically wounded character can be targeted, and it takes electric damage instead of status effects
`useCurrentBase`	yes/no	Base attack only: overrides the cost and flags of the container ability
`disableDamageReactionDuringAbilit`	yes/no	The soldier does not play a damage

y		reaction while performing the ability

Interface display

Field	Type	Meaning
showInPanel	yes/no	Show the button on the ability panel
panelViewType	list	Button style: Hidden, Simple, Attack, Targets, CustomAttack
showInWeaponPanel	yes/no	Show in the weapon panel
hideInPanelIfUnavailable	yes/no	Hide the button while the ability is unavailable
hideInInfoPanel	yes/no	Do not show in the soldier info window
hideInCharacterInfoWidget	yes/no	Do not show in the character widget
hideChances	yes/no	Do not show the hit chance
showForUse	yes/no	⚠ Show the use counter on the button
colorForUse	string	Color of this counter, for example #228CD9
highlightable	yes/no	⚠ Highlight the button
ignoreBuffsInPropertyView	yes/no	Ignore buffs when showing the ability stats

Animation, effects, sound

Field	Type	Meaning
abilityAnimationTrigger	string	Animation trigger: DoReload, DoPickUp, DoPutDown, DoToss, DoUnderbarrelLaunch, DoHealAlly, DoScan
abilityAnimationIndex	number	Gesture animation number: 1 — order, 2 — "rock", 3 — "rise"; the game uses 1–5
animationSpeed	number	Animation speed, 1 by default
vfxIdent	string	Visual effect, for example grenade_explosion

`vfxSoundIdent`	string	Sound of the visual effect
`abilityUseSound`, `abilitySoundEffect`, `abilityIntermediateSound`, `abilityEndSound`	string	Sounds of the ability start, middle and end
`abilityStartFoleySound`, `abilityIntermediateFoleySound`, `abilityEndFoleySound`	string	Soldier movement (foley) sounds
`abilityUseReplicaSound`, `abilityUsePreReplicaSound`	string	Soldier voice lines on use and before it
`playGenericAttackSound`	yes/no	Play the generic attack sound

A mod can't add new sounds, effects or animations; it can only reference existing identifiers.

The abilitySettings block

A set of optional nested blocks. Which of them an ability reads depends on its `abilityType`: use a game ability of the same type as a template.

Field or block	Contents	Meaning
`buffs`	string array	Buffs from `Buffs` that the ability grants
`overwatchBuffs`	string array	Buffs applied to the overwatch shot
`removeBuffs`	string array	Buffs the ability removes
`grantedAbilities`	string array	Abilities granted by this ability
`ProjectileKey`	string	Projectile type, for example `Projectile_01`
`weaponAbilitySettings`	`Accuracy`, `ShotCount`, `ShotDelay`, `AfterShotDelay`, `Ammo`, `AccuracyBonusAgainstDestructibleObjects`, `IsSpendAllAmmo`, `CanNotCritOrScratch`, `Repeat`	Shooting: accuracy bonus, number of shots, pause between shots and after a burst (seconds), ammo use,

		accuracy bonus against destructible objects, «uses the whole magazine», «no crits or grazes». `Repeat`: ⚠
`weaponDecorativeAbilitySettings`	the same fields	Shooting animation only, when it differs from the logic (area fire)
`damageAbilitySettings`	`minDamage`, `maxDamage`, `damageForArmor`, `ignoreArmor`, `minDamageForStruct`, `maxDamageForStruct`, `damageResistType`, `armorDamageType`, `selfDamage`, `ragdollImpulseForce`	The ability's own damage: damage range, armor damage, armor piercing, damage to buildings, damage type for resistances, self-damage, body knockback force. `armorDamageType`: ⚠
`healingAbilitySettings`	`addPoints`, `healthPercentValue`	Healing: hit points or a share of the maximum (0.5 = 50%)
`autoFirstAidAbilitySettings`	`incapacitatedTurnsUntilAutoFirstAid`	After how many turns an incapacitated soldier gets first aid automatically
`recoverArmorAbilitySettings`	`recoverArmorCount`, `usesArmorChangeAnimation`	How many armour points to restore; whether to play the plate change

		animation
actionPointsAbilitySettings	addActionPoints, type	How many points to add and which: Common, Move, Attack
aoeAbilitySettings	ShapeType, DamagePropagationType, AttackType, ActionType, AttackRangeDependsOnWeaponRange, WeaponRangeMultiplier, BlockedBySmoke, BlockedByFogOfWar, PenetratesHighObstacles, ExcludeDestructibleObjectsAroundAOEOrigin, UseRandomDirection, ForceDuckingIfSuppressed, AnimationDuration, ArcShootingParameters	Area effect. Shape: Cone, Sphere, Rect, Cylinder. Spread: Instantly, LeftToRight. Attack type: Overwatch, AttackAllCells, Suppression. Action: SetReaction, Shot, ShotWhileHasAmmo, UseItem. The range can depend on the weapon range with a multiplier; smoke, fog of war and high obstacles can block it
coneAbilitySettings	Width, MaxRange, FieldOfViewVisualMaxLength	Cone: width in degrees, range, drawn length
sphereAbilitySettings	MaxRange, Radius	Sphere: throw range and radius
cylinderAbilitySettings	MaxRange, Radius, Height, YCenterOffset	Cylinder: the same plus height
rectAbilitySettings	Width, Length	Rectangle
radiusAbilitySettings	radius	Radius of effect

		around the soldier
ballisticSingleCellAbilitySettings	MaxRange	Throw range to a single cell
areaAroundAbilitySettings	returnToStartCellOnComplete	Whether to return the soldier to the starting cell
targetFilteringAbilitySettings	CanAffectOwner, CanTargetEnemies, CanTargetAllies, CanTargetDestructibleObjects, CanTargetLoot, CanTargetMapObjective, CanTargetOnlyOverwatchers, IgnoreMachineTargets, IgnoreHumanTargets, CustomShape	Who the ability can affect or target
targetTypeFilteringAbilitySettings	ignoredUnitTypes	Unit classes the ability does not affect
sidestepsAbilitySettings	IgnoreSidesteps, UseBallisticAimTime	Ignore stepping out of cover; aiming time as for a throw
meleeSettings	canNotClinch, canNotExecute, doNotSpawnBlood	Melee: no clinch, no execution, no blood
deployableAbilitySettings	deployableId, turnToExplode, highlightPrefab, iconId, warningTextKey	Deployable object (mine, bomb): what is placed and after how many turns it triggers
additionalDeployableSettings	array of the same blocks	Additional objects, e.g. fire after the explosion
miningSettings	minesAmount	How many mines are

		placed
`underSelfSettings`	`associatedAbilityId`	Ability whose effect is created under the soldier
`droneSettings`	`removeAfterDeath`	Remove the drone after it dies
`durationSettings`	`abilityDuration`	Duration in turns
`turnIterableAbilitySettings`	`turns`	After how many turns the effect repeats
`activationChanceSettings`	`activationChance`	Trigger chance. Both formats occur in the game: `0.5` and `"30"`
`useAbilitySettings`	`useCount`	Number of uses
`experienceSettings`	`additionalExperiencePercent`	Experience bonus, 0.3 = +30%
`injuryDaySettings`	`deltaDaysToRecovery`	By how many days the injury recovery time changes
`statusEffectModifierSettings`	`statusEffectDurationModifiers`, `statusEffectBuffModifiers`	Changes to the duration and buffs of other status effects
`abilityVisualsSettings`	`meshIdent`	Item model in hand, for example `Grenade_Smoke_1`

Example: a new grenade based on an existing throw ability (fields not shown in the example must be copied from the template ability).

```
{
  "MyGrenade_throw": {
    "title": "ability/title/myGrenade",
    "description": "ability/description/myGrenade",
    "icon": "Ability/Grenade2",
    "abilityType": "Grenade",
    "abilityTargetType": "Ballistic",
    "actionPointPrice": 1,
    "charges": 2,
    "limited": true,
    "abilitySettings": {
      "damageAbilitySettings": { "minDamage": 5, "maxDamage": 8,
"damageResistType": "Explosion" },
      "sphereAbilitySettings": { "MaxRange": 11, "Radius": 3 }
    }
  }
}
```

2.2. StatusEffects

File `Configs/StatusEffects.json`. Object: the key is the effect identifier. The game has 61 effects: bleeding, burning, stun, disorientation, suppression, etc. An effect does nothing by itself: for a while it puts buffs from `Buffs` on the target.

Field	Type	Meaning
`name`	text key	Name
`description`	text key	Description
`icon`	sprite key	Icon
`type`	list	Effect category: `None`, `Stun`, `Disorientation`, `Suppression`, `Bleeding`, `Burning`, `Smoke`, `TearGas`, `PoisonGas`. Immunities and resistances work by category
`duration`	number	Duration in turns
`completeType`	list	How it ends: `Duration`: when the turns run out; `Custom`: on a condition in code (for example, leaving the smoke)
`stackType`	list	What happens when applied again: `DoNotStack`: the new one is not applied; `MaxDuration`: the longer duration stays; `StackDuration`: durations add up; `KeepBoth`: both stay with their own durations; `ReApplyBuffsIncremental`: the

		new one replaces the old one's buffs
`buffs`	array	The effect's buffs. Element: `buff` is an identifier from `Buffs`, `chance` is the chance to apply, from 0 to 1
`useBuffOn`	list	Who gets the buffs: `Target`: the target; `Someone`: ⚠ another participant
`positiveEffect`	yes/no	Positive effect (affects display)
`removeOnIncapacitate`	yes/no	Remove when the soldier is incapacitated
`throwType`	list	`Single` or `Multy`; `Multy` everywhere in the game ⚠
`visibleInGUI`	yes/no	Show the icon in the UI
`visibleOnCharacter`	yes/no	Show above the soldier
`showNamePopupOnStatusEffectThrow`	yes/no	Name popup on application
`showBuffPopups`	yes/no	Buff popups
`showType`	list	`None`, `Fix`, `Hidden` ⚠
`vfxIdents`	string array	Visual effects on the soldier
`entityEffectType`	list	Visual state of the object, for example `Burning`
`doNotLoad`	yes/no	The effect is not loaded by the game

Example: make burning last longer.

```
{
  "Burning": { "duration": 4 }
}
```

2.3. Buffs and debuffs

File `Configs/Buffs.json`. Object: the key is the buff identifier. The game has 283 buffs. A buff is one stat change or one rule; buffs are granted by abilities, status effects and items.

Field	Type	Meaning
`ident`	string	Identifier, same as the key

`buffType`	list	What exactly the buff changes. The list of values is in the documentation appendix
`value`	number	Amount in stat units: `20` for accuracy is +20, `-2` for mobility is minus 2 cells
`valuePercent`	number	Amount as a fraction: `0.3` = +30%, `-0.5` = −50%
`stackType`	list	Identical buffs: `DoNotStack`: the second is not applied; `MaxValue`: the larger stays; `StackValue`: `value`s add up (does not affect `valuePercent`); `ReApply`: the new one replaces the old one
`limitedActivationsPerTurn`, `maxActivationsPerTurn`	yes/no, number	Limit of activations per turn
`maxActivations`	number	Total triggers; `-1` means unlimited
`coverType`	list	For cover buffs: `Half` or `Full`
`statusEffectType`	list	Status effect category the buff relates to (resistance, immunity)
`damageTypes`	array	Damage types the buff reacts to: `None`, `Explosion`, `Fire`, `Shot`, `Bleeding`, `Knife`
`statusEffects`	string array	Status effects the buff applies
`selfBuffs`	string array	Buffs the owner receives
`otherTargetBuffs`	string array	Buffs another target receives
`requiredStatusEffectsIdents`, `requiredStatusEffectsTypes`	arrays	The buff works only if the target has these effects
`requiredIncapacitation`, `requiredStun`	yes/no	The buff works only against an incapacitated or stunned target

Nested blocks (needed only by buffs of the corresponding type)

Block	Fields	Meaning
`abilityBuffConfig`	`abilityIdent`, `additionalAbilityIdent`, `charges`,	For `AbilityBuff`: grants an ability or changes it:

	`isGeneralCharges`, `isShouldMultiply`, `isShouldAddOnce`, `rechargeDurationDelta`, `actionPointPrice`, `freeUsesForMission`, `availableAtZeroActionPoints`, `abilitySettings`	adds charges, shifts the recharge (`-1` is one turn faster), changes the cost, replaces parts of `abilitySettings`
`replaceAbilityBuffConfig`	`abilityIdent`, `replaceIdent`, `charges`	Replaces one ability with another
`actionPointsBuffConfig`	`addActionPoints`, `removeActionPoints`, `type`	Adds or removes points: `Common`, `Move`, `Attack`, `All`
`rechargeAbilitySettings`	`rechargeCount`	How many turns it cuts from ability recharges
`damageResistSettings`	`damageResistType`, `damageResistValue`	Resistance to a damage type
`armorFittingBuffConfig`	`hpBuffByArmorLevel`	Health bonus by armor level: an array, the element index is the armor level
`debuffOnAttack`	`debuffChance`, `selectOnlyOneDebuffWeighted`, `debuffs`	Chance to apply debuffs to the target on attack
`randomizedShotsSettings`	`shots`: `shotIdx`, `probability`	Chance of extra shots by shot number
`allyKilledBuffSettings`	`distance`	Radius within which an ally's death counts
`limitedBuffConfig`	`maxUses`	How many times the buff triggers

Example: a new accuracy buff.

```
{
  "accuracy_add_35": {
    "ident": "accuracy_add_35",
    "buffType": "Accuracy",
    "value": 35,
    "stackType": "StackValue"
  }
}
```

2.4. Attack: general hit rules

File `Configs/Attack.json`. A single object with global shooting coefficients. The hit chance is the soldier's accuracy plus the ability bonus, the range zone value and the height bonus, minus the target's defense and cover.

Field	Type	Meaning
`zones`	object	Range table by weapon type: `CloseRange`, `DefaultRange`, `SniperRange`, `MachinegunRange`, `TurretRange`. The value is an array of zones: `minDistance`, `maxDistance`, `aim` (accuracy bonus), `criticalChance` (crit chance bonus)
`accuracyBonusForHeight`	number	Accuracy bonus for shooting from height
`shotCountProtection`	number	Accuracy penalty for each next shot in a burst
`minDamageChance`	number	Lower bound of the hit chance, in percent
`scratchModifier`	number	Damage share on a "scratch": 0.3 = 30%
`halfCoverProtectionValue`, `fullCoverProtectionValue`, `flankedCoverProtectionValue`	number	Target protection behind half cover, full cover and when flanked
`halfCoverCritValue`, `fullCoverCritValue`, `flankedCoverCritValue`	number	Crit chance adjustment for the same cases
`angleCoverDebuff`	number	⚠ Cover weakening when shooting at an angle
`upStepDebuff`	number	⚠ Penalty for shooting upwards
`fullCoverSidestepPenalty`, `halfCoverSidestepCrChancePenalty`, `fullCoverSidestepCrChancePenalty`	number	Penalties for stepping out of cover to shoot
`targetingValue`	number	⚠ "Targeting" bonus
`directionalMissChance`, `directionalMissHitsToTargetsChances`	number, object	Chance that a miss goes sideways and hits a character, decor or an explosive object

2.5. Movement: movement cost

File `Configs/Movement.json`. How many mobility points one step between cells costs.

Field	Meaning
`transitionPriceForNormal`	Normal step
`transitionPriceForDiagonal`	Diagonal step
`transitionPriceForJumpOver`	Vaulting over an obstacle
`transitionPriceForLadder`	Ladder
`transitionPriceForPullUp`	Pulling up onto a ledge
`transitionPriceForHighJump`	Jump from height
`transitionPriceForAirTransition`	Air transition (for flyers)

3. Characters

3.1. CharactersLevelsData: levels and class progression

File `Configs/CharactersLevelsData.json`. Object: the key is the character class (appendix A.3); the game describes 24 classes. This sets what a soldier gets at each level and how much hiring costs.

Field	Type	Meaning
`ClassAbility`	string	Class ability from `Abilities`, available from level 1. An empty string means none
`LevelsData`	array	Levels. Merged by the `Level` field (see 1.3)

LevelsData element

Field	Type	Meaning
`Level`	number	Level number, the merge key (0–8 in the game)
`ExperienceAmount`	number	Experience required for the level
`LevelAbilities`	array	Abilities that can be obtained at this level. Replaced as a whole
`LevelAbilities[].Ident`	string	Ability from `Abilities`

`LevelAbilities[].ReqAbility`	string	Ability the soldier must have for this one to appear in the choice
`LevelAbilities[].ReplaceAbility`	string	Ability this one replaces
`SpecialAbility`	string	Ability granted at the level without a choice
`Price`	number	Hiring price of a soldier of this level
`PricePerMonth`	number	Monthly salary
`ArriveTimeDays`	number	Days it takes the soldier to arrive after hiring
`PropertyIncrease`	object	Stat bonus at this level: `DefaultHitPoints`, `BaseProtectionValue`, `Mobility`, `Accuracy`, `CriticalDamageChance`, `FieldOfView`, `FieldOfDetection`, `IncapacitationTurnBeforeDie`, `MeleeFactor`

Example: a level 1 Assault costs 5000 and gets +1 health.

```
{
  "Assault": {
    "LevelsData": [
      { "Level": 1, "Price": 5000, "PropertyIncrease": { "DefaultHitPoints": 1 }
}
    ]
  }
}
```

3.2. CharacterTemplates: character templates

File `Configs/CharacterTemplates.json`. Object: the key is the template identifier; the game has 293 of them: named mercenaries at different levels (`shilo8Character`), enemies (`NPC_TER_ShooterT0`), civilians, drones, turrets. A character on the map is created from a template; spawn sets and scenarios refer to templates.

Field	Type	Meaning
`characterClass`	list	Class (appendix A.3)
`level`	number	Level
`exp`	number	Starting experience
`modules`	object	What the character is built of: module type →

		preset identifier
`CharacterAbilities`	object	Abilities by level: `"1": ["Sprint", "Passive_Lucky"]`. A level's array is replaced entirely
`inventorySlots`	object	What is in the slots: slot → weapon or item identifier
`metaInfoIdent`	string	Biography and portrait identifier
`replicaIdent`	string	Voice line set
`genderType`	list	`Male` or `Female`
`uniqueCharacter`	yes/no	Named character
`canTakeCover`	yes/no	Can take cover
`price`, `sellPrice`, `pricePerMonth`	number	Hiring price, sell price (for drones), salary
`arriveTimeDays`	number	Days until arrival after hiring
`hideForShop`	yes/no	Do not show in hiring
`property`	object	Stats directly (see `PropertyIncrease` in 3.1). Used by generated mercenaries
`statusEffectTypeImmunities`	array	Immunity to effect categories: `["Stun", "Disorientation"]`
`statusEffectImmunities`	string array	Immunity to specific status effects
`buffTypeImmunities`	array	Immunity to buff types
`minSpreadDist`, `spreadMultiplier`	number	⚠ AI positioning settings: minimum distance between soldiers and its multiplier
`dlcRequirement`	string	DLC without which the character is unavailable
`isDrone`	yes/no	The character is a drone
`equipmentTier`	list	For drones: `Tier1`–`Tier3`

The modules block

Each line is «module type: preset identifier» from the matching `Configs/Modules/<Type>ModuleConfig.json` file (section 4). So to change a stat of one character,

you create a new preset in the module config and put it into the template.

Module	What it sets	Where the presets are
`HitPoints`	Health	`Modules/HitPointsModuleConfig`
`ActionPoints`	Action points	`Modules/ActionPointsModuleConfig`
`Movement`	Mobility	`Modules/MovementModuleConfig`
`Attack`	Accuracy and crit chance	`Modules/AttackModuleConfig`
`Melee`	Melee skill	`Modules/MeleeModuleConfig`
`CoverProtection`	Base defense (dodge)	`Modules/CoverProtectionModuleConfig`
`FieldOfView`	Vision	`Modules/FieldOfViewModuleConfig`
`ComplexDeath`	Critical wound and death rules	`Modules/DeathModuleConfig`
`BaseAbility`	Base set of abilities	`Modules/BaseAbilityModuleConfig`
`AIBrain`	AI behavior	`Modules/AIBrainModuleConfig`
`Weapon`	Weapon array. Replaced entirely	`Modules/WeaponModuleConfig`
`Item`	Item array. Replaced entirely	`Modules/ItemModuleConfig`
`UpdateMesh`	Appearance	`Modules/UpdateMeshConfig`
`Animation`, `Outline`, `CharacterUIPanel`, `BodyTransfer`	Animations, outline, overhead panel, carrying bodies	service

A module of a type the character didn't have is appended to the end of the list. The module order is stored in combat saves: a combat save made without such a mod may not fully restore with it.

Example: give level 8 Shilo 15 health. Two files are needed.

`Configs/Modules/HitPointsModuleConfig.json`:

```
{
  "HP_Example": { "DefaultHitPoints": 15 }
}
```

`Configs/CharacterTemplates.json`:

```
{
  "shilo8Character": {
    "modules": { "HitPoints": "HP_Example" }
  }
}
```

3.3. CharacterGeneration: mercenary generation

File `Configs/CharacterGeneration.json`. The rules for creating unnamed mercenaries for hire.

Field	Type	Meaning
`experienceMinLevel`, `experienceMaxLevel`	number	Level range of generated soldiers
`maleChance`	number	Chance of male gender, in percent
`defaultCharactersData`	object	A soldier blank for each class (`Newbie`, `Assault`, `Operator`, `Support`, `Sniper`, `Grenadier`). Same format as a template from 3.2
`characterPropertyGenerationData`	object	`minProperties` and `maxProperties` are the bounds of random stats: `DefaultHitPoints`, `BaseProtectionValue`, `Mobility`, `Accuracy`, `CriticalDamageChance`, `MeleeFactor`
`defaultBaseAbilityConfig`	string	Default base ability set
`baseAbilityConfigByClass`	object	Base ability set for particular classes
`birthplaces`	string array	Birthplaces for the biography
`raceByBirthplaceIndexes`	object	Which appearance variants fit a birthplace
`genderDependentInfos`	object	Models and text keys for each gender
`bioFormat`	array	Which parts and how many variants the biography text is built from

3.4. CharactersShop: hiring

File `Configs/CharactersShop.json`.

Field	Type	Meaning

`crewSalary`	number	Monthly pay per crew member
`firingTimeout`	number	Days a dismissed soldier does not appear in hiring
`uniqueCharacters`	object	Named mercenaries available for hire. `data` holds a character template (3.2)
`droneTemplates`	object	Drones available for purchase. Same format
`rotationConfigs`	object	List refresh rules for three groups: `Newbie`, `Experienced`, `Veteran`

rotationConfigs element

Field	Meaning
`startOfRotation`	Campaign day from which the group appears in hiring
`dayOfClear`	Every how many days the list is cleared
`dayOfUpdate`	Every how many days soldiers are added to the list
`maxAmount`	How many soldiers of the group can be in the list at once
`newAddProbability`	Chance of adding a new soldier on update, in percent
`clearRules`	How many to remove on a cleanup: `minCharactersForApply`: the rule applies from this many mercenaries; `clearAmount`: how many to remove

3.5. CharacterMetaBuffs: temporary bonuses in the hub

File `Configs/CharacterMetaBuffs.json`. The `buffs` list: stat bonuses a mercenary gets for a few days from training and random events.

Field	Meaning
`ident`	Identifier that events and training refer to: `HPBuff1`, `AccuracyDebuff10`
`additionCharacterProperty`	What is added: a set of stats, as in `PropertyIncrease` (3.1). A negative value is a penalty

The `buffs` array is replaced entirely: to add your own bonus, list all of the game's bonuses in the patch too.

3.6. CharacterMetaStatuses: mercenary states in the hub

File `Configs/CharacterMetaStatuses.json`. How long a wound, rest or training lasts and what speeds them up.

Field	Type	Meaning
`statuses`	object	State settings. Keys: `Ready`, `Relax`, `InjuredEasy`, `InjuredMedium`, `InjuredHard`, `OnMission`, `Arriving`, `InCity`, `Captured`, `Training`, `OnEvent`, `Repair`, `Dead`
`statuses.*.minDays`, `maxDays`	number	Duration range in days
`statuses.*.isPersistent`	yes/no	Status without a time limit
`statuses.*.applianceSettings.receivedDamagePercentMin`	number	For injuries: from what share of lost health this severity is assigned (0.4 = 40%)
`statuses.Training.rewardOnCompleteSettings.exp`	number	Experience for training
`statuses.Training.trainingBuffs`, `trainingBuffsDuration`	array, number	Bonuses from 3.5 to choose from and their duration in days
`injuryStatuses`	array	Which statuses count as an injury
`boosters`	object	Boosters: `Hospital`, `Reanimation`, `Saloon`, `SlotMachine`, `StimulatorDebuff`. Fields: `statuses`: what it affects; `boosterPercent`: how much it shortens the duration (0.5 = by half);

		`unlockedByFeature`: what unlocks it
`boosterModificators`	array	Temporary booster adjustments from events: `ident`, `boosterType`, `value` (fraction), `daysDuration`

4. Modules: stat presets

Folder `Configs/Modules`. Each file is an object: the key is the preset identifier, the value is its parameters. Presets are attached to characters through the template's `modules` block (3.2). Changing an existing preset affects everyone who uses it: `HP_9` is set on many characters. To change one character, create a new preset.

4.1. Simple modules

File	Field	Meaning
`HitPointsModuleConfig`	`DefaultHitPoints`	Health. Game presets: `HP_1` … `HP_60`
	`CanNotReceiveResilientPowerUp`	The unit does not get the health bonus from the "resilient enemies" setting
`ActionPointsModuleConfig`	`actionPoints`	Action points per turn, 2 by default
	`movePoints`	Extra points for movement only
	`attackPoints`	Extra points for attack only
`MovementModuleConfig`	`mobility`	How many cells a mercenary moves per action point. Presets `Speed_0` … `Speed_16`
	`movementType`	`Humanoid` or `Fly`
`AttackModuleConfig`	`accuracy`	Base accuracy
	`criticalDamageChance`	Base crit chance, in percent

`MeleeModuleConfig`	`meleeFactor`	Melee skill. Presets `Melee5` … `Melee100`
`CoverProtectionModuleConfig`	`BaseProtectionValue`	Base defense: subtracted from the chance to hit the mercenary. Presets `Evasion5` … `Evasion25`
`FieldOfViewModuleConfig`	`FielOfViewDistance`	View distance in cells (the field name is spelled as in the game)
	`FielOfDetectionDistance`	Detection distance
	`AlwaysVisible`	The unit is always visible
	`EnemyFogInvisibleRadius`	⚠ Radius within which an enemy in the fog of war stays hidden
`BaseAbilityModuleConfig`	`abilities`	The preset's base abilities: `["Attack", "Reload", "Overwatch", ...]`. The array is replaced entirely

4.2. DeathModuleConfig: critical wounds and death

Field	Type	Meaning
`IncapacitationTurnBeforeDie`	number	How many turns an incapacitated soldier survives without help
`IncapacitationTurnBeforeDie_Easy`, `IncapacitationTurnBeforeDie_Hardcore`	number	The same for easy and hardcore difficulty
`HitPointsAmountAddAfterResurect`	number	⚠ Health after the soldier is revived
`IncapacitationBuffs`	string array	Buffs on an incapacitated soldier
`isPossibleTransfer`	yes/no	The body can be carried
`ifDeathOnInit`	yes/no	The unit appears on the map already dead

deathChancesDefault, deathChancesMedium, deathChancesHigh	object	Death chances for the three lethality setting levels
…AlwaysIncapacitateOnZeroHP	yes/no	At zero health always incapacitation, never death
…BaseDeathChanceOnZeroHp	number	Base chance to die at once, in percent
…DeathChanceIncreasePerOverkillDamageUnit	number	Chance increase per unit of damage beyond health
…DeathChanceIncreasePerHeavyWoundBuff	string	Buff that raises the death chance for each previous incapacitation
IncapacitatedAnimationBool, DeadAnimationBool	string	Animator parameters; no need to change

4.3. Map effect modules

File	Field	Meaning
SmokeGrenadeModuleConfig	turnToDisappear	How many turns the smoke lasts
	visibleRadius	Cloud radius
	smokePenalty	Accuracy penalty when shooting through smoke
	blockTypesInSmoke	Ability types unavailable in smoke
	availableOnScan	Ability types that are available again against a scanned target
FireModuleConfig	turnToDisappear	How many turns the fire burns
	radius	Fire radius
	chanceToBurn	Chance to set a soldier on fire, in percent
MineModuleConfig	turnToDisappear	⚠ Turns after which the object disappears

	`isInvisible`, `visibilityRadius`	Hidden mine and the radius from which it is visible
	`aoeAbilitySettings`, `sphereAbilitySettings`	Explosion shape and radius (as in 2.1)
`ScanGrenadeModuleConfig`	`turnToDisappear`, `radius`	Scanner duration and radius
`VolitionModuleConfig`	`baseValue`, `repairValue`	Starting volition and the value it recovers to
	`missShotValue`, `damageValue`, `alliesDeadValue`, `teamDeadValue`, `civilianDeadValue`	Volition change for a miss, a wound, death of an ally, a squad member, a civilian
	`effects`	Status effects when willpower drops to zero; `panic` in the game

4.4. AIBrainModuleConfig: AI behavior

An object of behavior presets for enemies and allies (`TER_Shooter`, `NPC_Civilian`, etc.). A preset defines the behavior tree and position-choice weights. This document does not cover the fields: to change a unit's behavior, set a different existing `AIBrain` preset in its template.

5. Equipment

5.1. Modules/WeaponModuleConfig: weapons

File `Configs/Modules/WeaponModuleConfig.json`. Object: the key is the weapon identifier; the game has 199 entries (including weapons of enemies, drones and turrets).

Field	Type	Meaning
`ident`	string	Identifier, same as the key
`itemName`, `itemDescription`, `itemDescriptionLoot`	text key	Name, description and description for the loot screen
`itemIcon`, `iconSmall`, `combatItemIcon`, `emptyIcon`	sprite key	Storage icon, small icon, combat icon and the empty slot outline
`tier`	list	Equipment tier: `Tier1`, `Tier2`, `Tier3`
`price`	number	Purchase price
`sellPrice`	number	Sell price. 0 — cannot be sold

`weaponClass`	list	Weapon class (appendix A.4)
`inventorySlot`	array	Which slot it goes into: `MainWeapon`, `SpecialSlot`, `EquipmentSlot`, `Armor`
`availableCharacterClass`	array	Classes that can carry the weapon
`abilities`	string array	Abilities the weapon grants: attack, overwatch and so on. Replaced as a whole
`minDamage`, `maxDamage`	number	Damage range
`damageForArmor`	number	Armour damage
`ignoreArmor`	yes/no	Damage bypasses armour
`damageForStruct`	number	Damage to structures and destructible objects
`damageResistType`	list	Damage type for resistances: `Kinetic`, `Impact`, `Fire`, `Electric`, `Explosion`
`criticalDamageChance`	number	Crit chance bonus, in percent
`criticalDamageMod`	number	Extra damage on a crit
`charges`	number	Rounds in the magazine
`doNotShowCharges`	yes/no	Do not show the ammo counter
`range`	number	Maximum firing range
`idealRange`	number	Distance the AI tries to keep with this weapon
`weaponRange`	list	Range table from `Attack.zones`: `CloseRange`, `DefaultRange`, `SniperRange`, `MachinegunRange`, `TurretRange`
`missShotSettings`	object	Where a miss goes: `radius` is the scatter radius around the target. The fields `nothing`, `enemy`, `allies`, `barrel` are not filled in the game ⚠
`bulkiness`	number	⚠ Bulkiness. 0 or 2 in the game

`shootSpeed`	number	Projectile flight speed (visual)
`ragdollImpulseForce`, `ignoreHeadshotRagdoll`	number, yes/no	Ragdoll impulse on a kill
`isDual`	yes/no	Dual weapon
`modelKey`	string	Weapon model
`animatorId`	string	Animation set; `Base` everywhere in the game
`abilityUseSound`, `abilityReloadSound`, `abilityReloadFoleySound`, `startShotSound`, `stopShotSound`	string	Shot and reload sounds
`showInUI`	yes/no	Show in storage and in the shop
`stackableInStorage`	yes/no	Identical copies stack into one cell
`itemCharacteristicsType`	string	Which set of stats to show on the item card: `weapon`, `knife`, `taser`, `grenadelauncher`, `climbinghook`, `droneWeapon`
`faction`, `reputation`	string, number	⚠ Faction and reputation. The fields exist in the game file, but their use by the game is not confirmed

Example: change an existing rifle.

```
{
  "SniperRifleT1": {
    "price": 777,
    "sellPrice": 333,
    "minDamage": 4,
    "maxDamage": 6
  }
}
```

A new weapon is a new entry with the full set of fields (the easiest way is to copy a similar weapon's entry and change the key, `ident` and the values you need). To make it reach the player, add it to `InventoryConfig`, a character template, loot or the black market.

5.2. Modules/ItemModuleConfig: items and armor

File `Configs/Modules/ItemModuleConfig.json`. Object: the key is the item identifier; the game has 143 entries: armor, grenades, medkits, mines, gadgets.

Field	Type	Meaning
`ident`	string	Identifier, same as the key
`itemName`, `itemDescription`, `itemDescriptionLoot`	text key	Name and descriptions
`itemIcon`, `iconSmall`	sprite key	Icons
`itemType`	list	`Armor`, `Grenade`, `Medkit`, `Technique`, `Consumable`, `Trophy`
`tier`	list	`Tier1`–`Tier3`
`price`, `sellPrice`	number	Purchase and sell price
`inventorySlot`	array	Slot: `Armor`, `EquipmentSlot`, `SpecialSlot`
`openedInventorySlots`	string array	Which extra slots the item unlocks: `EquipmentSlot2`, `EquipmentSlot3`
`availableCharacterClass`	array	Classes the item is available to
`charges`	number	Number of charges (uses per mission)
`abilitiesSettings`	array	Item abilities. Element: `abilityName` is an ability from `Abilities`, `abilitySettings` are its parameters for this item (blocks from 2.1). Replaced entirely
`dependAbilities`	string array	Soldier abilities without which the item is unavailable
`armorSettings`	object	Armour parameters
`characterProperty`	object	Stat bonus for the owner
`maxStoredAmount`	number	How many can be kept in storage
`showInUI`, `stackableInStorage`	yes/no	Display in storage and stacking into one cell
`itemCharacteristicsType`	string	Stat set on the item card: `armor`, `grenade`, `smokegrenade`, `medicine`, etc. (the full list is the keys of `characteristicsByItem` in `HoldConfig`)
`faction`, `reputation`	string, number	⚠ Faction and reputation. The fields exist in the game file, but their use by the game is not

		confirmed
`abilityUseSound`, `triggerOverwatch`	string, yes/no	Use sound; whether using it triggers overwatch fire

The armorSettings block

Field	Meaning
`armorPoints`	Armour points
`armorUnitValue`	⚠ How much damage one armour point absorbs
`armorLevel`	Armour level: "armour fitting" buffs depend on it
`maleArmorVisualTier`, `femaleArmorVisualTier`	Appearance number on the male and female model
`damageResist`	Resistance by damage type: `{ "Kinetic": 0.0 }`

Example: strengthen tier 1 armor.

```
{
  "SpartaArmorT1": {
    "price": 1200,
    "armorSettings": { "armorPoints": 2 }
  }
}
```

5.3. InventoryConfig: slots and starting equipment

File `Configs/InventoryConfig.json`.

Field	Type	Meaning
`defaultInventorySlots`	string array	Slots every soldier has
`inventorySlots`	object	Slot descriptions: `ident` and `type` (`MainWeapon`, `Armor`, `SpecialSlot`, `EquipmentSlot`)
`defaultInventory`	object	Default equipment for a class: slot → weapon or item identifier

Example: an Assault gets tier 1 armor by default.

```
{
  "defaultInventory": {
    "Assault": { "Armor": "SpartaArmorT1" }
  }
```

```
}
```

The item must suit the class: the mercenary's class must be in the item's `availableCharacterClass`.

5.4. HoldConfig: the storage

File `Configs/HoldConfig.json`.

Field	Meaning
`characteristicsByItem`	Which stat lines to show on the card for each `itemCharacteristicsType`
`itemCharacteristics`	Description of the stat rows themselves (name, icon)
`contractOpenDiscount`, `contractOpenDiscountDuration`	Discount on an item after its contract is unlocked and its duration in days
`periodicDiscountsCount`, `periodicDiscount`, `periodicDiscountDuration`	Periodic discounts: how many items, size, duration
`aiContractDiscountMultiplier`	⚠ Contract discount multiplier

All discounts in the game are currently zero.

5.5. Models: weapon models

A mod can add a weapon with its own 3D model or replace the model of an existing weapon (a reskin). The model goes into the mod's `Models` folder, and the game loads it by itself when the mod is enabled. Formats: binary FBX 7.x, glTF 2.0 or GLB, OBJ; ASCII FBX is not supported. The model is not bound to the character skeleton: it is held rigidly in the hand like the game's weapons, and bone weights, if any, are ignored.

```
my_mod/
  Models/
    MyRifle.fbx          model (or .obj, .glb, .gltf)
    MyRifle.json         model parameters, required
    MyRifle_albedo.png   textures
    MyRifle_normal.png
    MyRifle_mask.png
  Configs/Modules/WeaponModuleConfig.json
  Sprites/Weapon/MyRifleT1.png

  Localizations/English
```

Model key. The model file name without the extension is the model key; it goes into the `modelKey` field of the weapon entry (5.1). A mod model with the key of an existing weapon (for example, `Models/AR_1.fbx` and `Models/AR_1.json`) replaces that weapon's look, and the weapon

config doesn't need to change. If the mod model fails to build and the game has such a key, the weapon keeps the game model and a warning is written to the log. If two mods bring a model with the same key, the mod lower in the list wins.

Model requirements. Units are meters, the barrel points forward (+Z), the grip down (−Y). The fit in the hand is set by the orientation of the meshes inside the model: the root pose is reset when attached, so rotate the geometry, not the root. Points on the weapon are set with empty named nodes:

Node	Purpose	Required
`actionPoint`	Muzzle: projectiles and the flash come out of here	yes; without it projectiles fly from the model root
`magazine`	Magazine that is taken out on reload	no
`magazineInstance`	Copy of the magazine for the hand during reload	no
`laserPoint`	Laser point	no
`cartridgeCasesParent`	Point where cartridge cases are ejected	no

OBJ carries nodes less well than FBX and glTF, so a model with points is easier to make in FBX or glTF.

Model parameters are the `Models/<key>.json` file next to the model. The required `weaponClass` field must match the `weaponClass` of the weapon entry with this `modelKey`, otherwise the game won't attach the weapon to the character.

Field	Meaning	Default
`weaponClass`	Weapon class (appendix A.4)	required
`animationClass`	Character animation set	same as `weaponClass`
`handIKClass`	Finger pose on the weapon	same as `weaponClass`
`weaponLength`	Weapon length in meters, used by the AI and animations	`1.0`
`splashIdent`	Muzzle flash effect: `assaultRifle_muzzle_1`, `LMG_muzzle_1`, `pistol_muzzle_1` …	no flash
`cartridgeCasesEffectIdent`	Cartridge case effect: `Ak47_ShellEmitter`, `ShootGun_ShellEmitter`	no cases

`actionPoint`, `magazine`, `magazineInstance`, `laserPoint`, `cartridgeCasesParent`	Node names, if the model names them differently	the names from the table above
`holsterOffsets`	Weapon position in a holster: `OnBack`, `OnHipRight`, `OnHipLeft` → `position` [x, y, z]	as in the hand
`textures`	Texture paths from the `Models` folder: `albedo`, `normal`, `mask`	by file names

Most weapon classes have animation sets and finger poses of the same name, so `animationClass` and `handIKClass` can be omitted. For classes without such a set (for example, `ClimbingHook`), set them explicitly.

```
{
  "weaponClass": "AutomaticRifle",
  "weaponLength": 0.85,
  "splashIdent": "assaultRifle_muzzle_1",
  "cartridgeCasesEffectIdent": "Ak47_ShellEmitter",
  "holsterOffsets": {
    "OnBack": { "position": [0.0, 0.1, -0.1] }
  },
  "textures": { "albedo": "MyRifle_albedo.png", "normal": null, "mask": null }
}
```

Textures. Materials are not taken from the model file: the game puts an HDRP/Lit material on every slot and applies the textures from the `Models` folder: `<key>_albedo.png` (or `_basecolor`) for color, `<key>_normal.png` for normals, `<key>_mask.png` for the HDRP mask (metallic, smoothness, AO). Formats: png and jpg. One set of textures is used for all slots of the model: different textures on different slots are not supported. Without textures the model is gray.

A whole new weapon. Besides the model, the mod brings:

- an entry in `Configs/Modules/WeaponModuleConfig.json` (5.1): `modelKey` is the model key, `weaponClass` the same as in the model parameters, `abilities` existing attack abilities, `showInUI: true` makes the weapon appear in the shop and the storage;
- an icon `Sprites/Weapon/<ident>.png` (the `itemIcon` field);
- the name and description in `Localizations/English`: `weapon/name/<ident>`, `weapon/description/<ident>`.

Example of a new weapon entry:

```
{
  "MyRifleT1": {
    "ident": "MyRifleT1",
    "tier": "Tier1",
    "itemName": "weapon/name/MyRifleT1",
    "itemDescription": "weapon/description/MyRifleT1",
    "itemIcon": "Weapon/MyRifleT1",
    "combatItemIcon": "Weapon/MyRifleT1",
    "price": 500,
    "sellPrice": 250,
```

```
    "animatorId": "Base",
    "itemCharacteristicsType": "weapon",
    "inventorySlot": ["MainWeapon"],
    "weaponClass": "AutomaticRifle",
    "availableCharacterClass": ["Newbie", "Operator", "Grenadier"],
    "charges": 4,
    "range": 18.0,
    "idealRange": 6,
    "minDamage": 1,
    "maxDamage": 2,
    "damageForArmor": 2,
    "abilities": ["AssaultRifle_Attack", "Overwatch_Basic"],
    "modelKey": "MyRifle",
    "weaponRange": "DefaultRange",
    "abilityUseSound": "WEAPON_ak12_shot_one",
    "showInUI": true,
    "stackableInStorage": true
  }
}
```

The other fields of the entry (5.1) are easiest to copy from a similar game weapon. The weapon can be given to a mercenary through the shop or a character template (3.2; the `Weapon` array is replaced entirely).

Limitations:

- Sounds, animations, effects and projectiles must be existing ones: `abilityUseSound` and the other sound fields, `animationClass`, `splashIdent`, `cartridgeCasesEffectIdent` take the game's identifiers.
- Mod weapons are stored in saves. If such a save is loaded without the mod, the game warns you and removes the unknown weapon.
- An error in the model doesn't crash the game: the log shows `[Mods] Can't assemble weapon model …`, and the weapon uses the game model or doesn't appear.

Checking. Enable the mod and look at the weapon in combat: the barrel points forward, projectiles and the flash come out of the muzzle, the magazine is taken out on reload, the weapon sits straight in the holster (otherwise adjust `holsterOffsets`). The log should have no `[Mods]` errors about the model.

6. Economy and hub

6.1. Currency: resources

File `Configs/Currency.json`. Three resources: `money`, `intel`, `crew`.

Field	Type	Meaning
`canBeNegative`	yes/no	Whether the resource can go negative
`iconId`	sprite key	Icon
`color`	string	Color in the interface, for example `#ECC026`

A mod can't add a new resource: the list is hardcoded in the game.

6.2. Reward format

The same format is used in loot (6.3) and mission rewards (7.5). The kind of reward is determined by which field it has.

Field	Type	Meaning
`item`	string	Item reward: identifier of a trophy, weapon or item
`currency`	string	Resource reward: `money` or `intel`
`experience`	`true`	Reward — experience for soldiers
`reputation`	string	Reward: reputation with the given faction
`profileSpecificReputation`	`true`	Reputation with the mission's client faction
`daamatPowerReduced`	`true`	Daamat power reduction
`valueRange`	`[min, max]`	Quantity range
`value`	number	Exact amount, when a range is not needed
`chance`	number	Drop chance, in percent
`isVisible`	yes/no	Show the reward before the mission. A hidden reward with a chance below 100 counts as a trophy
`ignoreMultiplication`	yes/no	Do not apply difficulty multipliers
`daamatPowerRequirement`	`[min, max]`	The reward drops only when Daamat power is in this range
`choiceRequirement`	string	The reward drops only after the given story choice

6.3. LootConfig: mission loot

File `Configs/LootConfig.json`. Loot is carried by individual enemies, the «carriers».

Field	Type	Meaning
`maxItemsPerBearer`	number	Maximum items per loot bearer
`lootBarrersChancesPerDifficulty`	object	Carrier spawn chances

		by mission difficulty: `"Normal": [100, 40, 25]`: the first carrier with a 100% chance, the second 40%, the third 25%
`reinforcementLootBearerChance`	number	Chance that a reinforcement enemy becomes a loot bearer, in percent
`difficultySettingsAdditionalBearerChances`	object	Chance bonus for enabled difficulty settings: `StrongDaamat`, `ResilientEnemies`, `AccurateEnemies`
`difficultySettingsAdditionalBearerChancesForSwitches`	object	The same for switch settings: an array by switch position
`lootByRegion`	object	Loot tables by region: `DefaultRegion`, `DLC1`, `DLC2`, `DLC3`. The value is an array of rewards (6.2). The array is replaced entirely
`collectables`	object	Collectible trophies: `collectableIdent`, `tier`, `itemName`, `itemDescription`, `icon`, `iconSmall`
`lootCollectedKey`	text key	Loot pickup message

Example: change the main region's loot table. The array is replaced entirely, so the patch must contain all the rows you want to keep; only two are shown here.

```
{
  "lootByRegion": {
    "DefaultRegion": [
      { "item": "equipment", "daamatPowerRequirement": [0, 100],
"ignoreMultiplication": true, "valueRange": [1, 3], "chance": 50 },
      { "item": "SniperRiflePT2_loot", "daamatPowerRequirement": [30, 100],
"valueRange": [1, 1], "chance": 5 }
    ]
  }
```

```
}
```

6.4. Missions/BlackMarketConfig: the black market

File `Configs/Missions/BlackMarketConfig.json`.

Field	Type	Meaning
`maxItemCount`	number	How many goods in one market visit
`timeToLive`	number	How many days the market stays open
`cooldown`	number	How many days until the next appearance
`unlockedItemSpawnChance`	number	Chance of each available good to get into the visit's assortment, in percent
`discountedItemsCount`, `discountMultiplier`	number	How many goods are discounted and the discount size (0.5 = 50%)
`markupItemsCount`, `markupMultiplier`	number	How many goods have a markup and its size
`greetingPhrases`, `closingPhrases`	string array	Trader voice lines
`productsData`	object	Assortment: the key is the good's identifier

productsData element

Field	Meaning
`itemId`	Weapon or item identifier
`minValue`, `maxValue`	Quantity range on sale
`choiceForRollUnlock`	Story choice after which the good starts appearing
`choiceForUnlock`	Choice after which the good is available
`choiceForReward`	Choice recorded after the purchase

Example: add a weapon from the mod to the stock.

```
{
  "productsData": {
    "MyRifle": { "itemId": "MyRifle", "minValue": 1, "maxValue": 1 }
  }
```

```
}
```

6.5. Contracts

File `Configs/Contracts.json`. Contracts unlock equipment and features for money and time.

Field	Type	Meaning
`contractConfigs`	object	Contracts: the key is the identifier
`companies`	object	Contractor companies: `name`, `description` (text keys), `icon`
`fastCompletePricePerDay`	number	Speed-up price per remaining day
`discountRate`	number	⚠ Discount size, in percent
`discounts`	array	Discounts from built compartments: `value` is a percentage, `discountType` is `Price` or `Time`, `affectedFeatures` is which features enable it
`hiddenStateShowTimeFromDays`	number	How many days before completion the remaining time is shown

contractConfigs element

Field	Meaning
`title`, `description`	Text keys
`company`	Company from `companies`
`price`	Price
`daysToUnlock`	Completion time in days
`visibilityRequirements`	When the contract is visible: `days`: from which day; `choiceForUnlock`: after which decision
`availabilityRequirements`	What is required to take the contract
`reward`	What the contract gives, for example `unlockedEquipIdent`, the identifier of the unlocked equipment

6.6. CompartmentConfig: base compartments

File `Configs/CompartmentConfig.json`. Object: the key is the compartment identifier (`hospital`, `saloon`, `barrack2`, etc.); the game has 22 compartments.

Field	Type	Meaning
`title`, `additionalTitle`, `description`, `bonusName`	text key	Texts
`mainImage`, `bonusIcon`	sprite key	Images
`buildDuration`	number	Build time in days
`reqs.money`, `reqs.crew`	number	Price and required crew
`reqs.contracts`	string array	Contracts that must be completed
`reqs.compartmentComplete`	string array	Compartments that must be built
`reqs.choices`	string array	Required story choices
`rewards.crew`, `rewards.characters`, `rewards.money`	number	How many crew slots, soldier slots and how much money the compartment gives
`rewards.contractsUnlock`	string array	Contracts it unlocks
`rewards.featuresUnlock`	string array	Features it unlocks
`rewards.equipmentUnlockIdents`	string array	Equipment or drones it unlocks
`resourceInfoType`, `bonusPercent`	list, yes/no	How to show the bonus in the UI

6.7. Factions

File `Configs/Factions.json`.

Field	Type	Meaning
`reqsForUnion`	object	What an alliance requires: keys `"0"`, `"1"`, `"2"` are the first, second and third alliance in order; values are `reputation` and `intel`
`factions`	object	Factions: `Sparta`, `Daamat`, `Daamat_x_G9`, `G9`, `Armada`, `Orphans`, `Wolves`, `Shield`, `TsaiShen`, `Front`, `Unknown`

factions element

Field	Meaning
`factionIdent`	Identifier

`factionTitle`, `factionDescription`, `leaderTitle`, `leaderDescription`	Text keys
`factionIcon`, `leaderIcon`	Sprite keys
`factionColor`	Colour in the UI
`EnemySubFaction`	Which enemy side it corresponds to
`unlockChoice`, `unionChoice`	Choices recorded on first contact and on alliance
`factionSquads`	Allied faction squads: an array of line-ups, a line-up is an array of character templates

6.8. DaamatPowerConfig: Daamat power

File `Configs/DaamatPowerConfig.json`. Daamat power (0–100) determines which enemy squads appear on missions.

Field	Meaning
`powerPerMonth`	Power growth per month
`powerIncrementPerMission`	Growth per mission
`powerGrades`	Minimum power by date: `"2035-04-30": 20`
`strategicOffensive`	Strategic offensive: `maxMissionCount`, `maxAllyMissionCount`: how many missions; `maxDays`: the duration

6.9. HubTimeConfig: time speed

File `Configs/HubTimeConfig.json`. `defaultTimeSpeed` and `fastTimeSpeed` are the speed of time in the hub at normal and fast speed. ⚠ Judging by the game's values (1800 and 28800), this is the number of game seconds per real second.

7. Missions

7.1. Missions/ActiveMissionsData and PassiveMissionsData: mission types

Two files of the same format. Object: the key is the mission type (appendix A.5). An active mission is a battle on a map; a passive one sends mercenaries away for a period of time.

Field	Type	Meaning
`descriptionKeys`, `objectivesKeys`	array of text keys	Description and objective variants

`infoBlockImages`, `levelImages`, `missionIcon`	sprite keys	Images
`mapIds`	string array	Maps the mission can take place on (active missions)
`additionalMapIds`	string array	Additional maps
`maxUnits`	number	How many soldiers can be taken
`target`	list	Enemy: `Daamat`, `Armada`, `Orphans`, `Wolves`, etc.
`region`	list	Region: `DefaultRegion`, `DLC1`, `DLC2`, `DLC3`
`timeToLive`	number	⚠ Present in the game file but not read by the game: the duration is set by `liveTime` in `difficulties`
`canUseRocket`	yes/no	Rocket strike available
`isTaserRequired`	yes/no	Taser required
`hiddenCivilianCount`	`[min, max]`	How many civilians are hidden
`requiredClass`	list	Class without which the mission cannot be started (passive missions)
`rescueMissionType`	list	Type of the rescue mission that appears after a passive mission fails
`rescueActiveChance`	number	⚠ Chance of a rescue mission appearing; use by the game is not confirmed
`difficulties`	object	Parameters by difficulty: `Normal`, `Hard`, `Nightmare`

difficulties element

Field	Meaning
`rewardMultiplierActive`, `rewardMultiplierPassive`	Reward multiplier
`difficultyMultiplierActive`	Enemy strength multiplier
`failMultiplierPassive`	⚠ Fail chance multiplier. Present in the game file, use by the game is not confirmed

`searchTime`	`[min, max]` days before the mission appears on the map
`liveTime`	`[min, max]` days the mission stays available
`timeToComplete`	`[min, max]` days to complete (passive)
`failChance`	`[min, max]` failure chance in percent (passive)

7.2. Missions/MissionsControllerConfig: general mission rules

Field	Meaning
`activeSpawnCooldown`, `passiveSpawnCooldown`	Every how many days an active and a passive mission appear
`activeTypeRollChances`, `passiveTypeRollChances`	Mission type weights. Separate tables ending in `DLC1`, `DLC2`, `DLC3` are for the expansion regions
`missionDifficultyChances`	Difficulty chances: `Normal`, `Hard`, `Nightmare`
`rewardMonthlyCoefficient`	Reward growth per campaign month (0.05 = +5%)
`monthlyPassiveFailIncrementCoefficient`, `maxMonthNumberForFailIncrement`	Monthly growth of the passive mission fail chance and the month after which it stops growing
`teamLvlFailDecrementPercent`	How much the team level lowers the fail chance
`compartmentPassiveBonus`	Compartment bonus: lower passive mission fail chance
`cityCompartmentBuff`	Compartment bonus: extra intel from cities
`cityLockDuration`	For how many days a city is locked
`cityDifficultyProfileChangeIntervalDays`	Every how many days cities change their difficulty profile
`citySpawnedMissionMoneyAddition`	Money bonus for a mission found in a city
`rocketLaunchTime`	Rocket flight time

`factionSquadCooldownDays`	Pause between faction squad calls
`rescueRollChances`	Chances of the outcomes of a failed passive mission
`spawnByCityBaseCategoryChance`	Chances of the category of a mission spawned from a city
`mainGameRegionChances`, `endGameRegionChances`	Region weights before and after the finale
`factionsWithoutDaamatPowerIncrease`	Enemies whose missions do not increase Daamat power

7.3. Missions/Cities and CityDifficultyProfiles: cities

`Cities` is an object: the key is the zone (`Zone1` … `Zone9`, `Zone Ports`). Fields: `zoneId`, `name`, `country`, `population`, `description`, `completeDescription` (text keys), `image` (sprite key), `maxUnits`: how many mercenaries can be sent.

`CityDifficultyProfiles` holds three city reconnaissance profiles: `Normal`, `Hard`, `Nightmare`.

Field	Meaning
`profileChance`	Profile selection weight
`searchTimeRange`	`[min, max]` days of reconnaissance
`failChanceRangePercent`	`[min, max]` failure chance
`failChanceIncrement`	Fail chance growth per cycle
`rewardMultiplier`	Reward multiplier
`activeTypeRollChances`, `passiveTypeRollChances`, `activeTypeRollChancesDLC1`	Weights of the mission types found by reconnaissance
`activeDifficultiesChances`, `passiveDifficultiesChances`	Difficulty chances of the found mission

7.4. Missions/StoryMissions: story missions

An object with two lists: `MainStory` (battles) and `MainStoryPassive`. The arrays are replaced entirely, so to edit one mission the patch must carry the whole list.

Field	Meaning
`mapId`	Map

`nameKey`, `missionTypeKey`, `descriptionKey`, `objectivesKeys`	Text keys
`levelImage`, `infoBlockImage`	Images
`maxUnits`	Squad size
`target`, `client`, `ally`	Enemy, client and ally
`DaamatPowerSetup`	Daamat power used to pick enemies
`requiredChoices`	Choices, one of which is required to start
`showDelayDays`, `timeToLiveDays`	After how many days it appears and how long it stays available
`rewardMultiplier`	Experience multiplier
`isTaserRequired`, `canUseRocket`	Taser required; rocket available

7.5. Missions/RewardsConfig: rewards

Field	Type	Meaning
`killExperience`	number	Experience per kill
`intelPerCapturedEnemy`	number	Intel per captured enemy
`maximumTrophiesPerMission`	number	Maximum trophies per mission
`difficultyCurrencyMultipliers`	array of three numbers	Money reward multipliers for the three positions of the «reward size» setting; `[1.4, 1, 0.6]` in the game
`missionRewards`	object	Rewards by mission type: type → array of rewards (6.2)
`storyMissionsRewards`	object	Story mission rewards: map identifier → array of rewards
`choiceRewards`	object	Rewards and penalties for story decisions: `choiceRewards`, `choicePenalties`
`activeFactionRewards`, `activeFactionPenalties`, `passiveFactionRewards`, `passiveFactionPenalties`	reward arrays	Reputation with the client for success and failure

Example: increase the money for a clearing mission. The reward array is replaced entirely, so the experience row is repeated as well.

```
{
  "missionRewards": {
    "Sweep": [
      { "experience": true, "isVisible": false, "valueRange": [25, 25], "chance":
100 },
      { "currency": "money", "isVisible": true, "valueRange": [8000, 10000],
"chance": 100 }
    ]
  }
}
```

7.6. Other mission configs

File	What it contains
`Missions/FactionProfilesConfig`	Profiles `Profile0`–`Profile6`: who fights whom at different stages of the campaign. `conflictTypeChances`: weights of the options «against Daamat», «faction against Daamat», «faction against faction»; `missionProfiles`: for each mission type, the lists `clients`, `targets`, `allies`
`Missions/ConnectedMissionsData`	Double story missions: `firstMapId`, `secondMapId`, `minUnitsToStart` (minimum squad), `allyFaction`
`Missions/AutoFightDataConfig`	Auto-battle: `enemiesCountGrades`: enemy count thresholds; `injuryChanceGrades` and `deathChanceGrades`: injury and death chances at each threshold; `daamatPowerGradeIncrement`: the Daamat power step
`Missions/ObjectSearchConfig`	Settings for object search missions

8. Enemies

8.1. AreaSpawnEnemyPowerConfig: who appears on the map

File `Configs/AreaSpawnEnemyPowerConfig.json`. An array of rules merged by four key fields (see 1.3). For a spawn point on the map, the game finds a rule by point type, Daamat power and enemy, and picks a set from it.

Field	Type	Meaning
`SpawnType`	string, key	Spawn point type on the map: `basic`, `big`, `support`, `melee`, `sniper`, `reinforcement` and others

`PowerMin`, `PowerMax`	number, key	Daamat power range
`FactionName`	list, key	Enemy
`AreaSpawnPacks`	array	Sets to choose from: `AreaSpawnPackID` is a set from 8.2, `Chance` is a weight from 0 to 1. Replaced entirely

Example: at low Daamat power, regular points produce a large group.

```
[
  {
    "SpawnType": "basic",
    "PowerMin": 0,
    "PowerMax": 9,
    "FactionName": "Daamat",
    "AreaSpawnPacks": [
      { "AreaSpawnPackID": "big_0_D_1", "Chance": 1.0 }
    ]
  }
]
```

In the game file one key combination occurs twice (`reinforcement`, 90–100, `Daamat_x_G9`): the patch goes into the first of the two entries.

8.2. AreaSpawnPackConfig: enemy sets

File `Configs/AreaSpawnPackConfig.json`. Object: the key is the set identifier; the game has 756 sets.

Field	Type	Meaning
`MaxSpawnCount`	number	How many units from the main list are created
`ForceSpawnTemplateNames`	string array	Main list: character templates from 3.2. Taken in order, cyclically
`EnemiesAmountNormal`	string array	Who is added on top of the main list when the enemy count setting is in the second position
`EnemiesAmountHard`	string array	Who is added when the same setting is in the third position
`NightmareMissionDifficultyTemplateNames`	string array	Who is added on `Nightmare` difficulty missions
`TeamToSpawnInto`	list	Side: `Enemy`, `Sparta`, `Allies`, `Civilian`, `None`

`Shuffle`	yes/no	Shuffle the order

Example: a new set of three shooters.

```
{
  "my_pack_three_shooters": {
    "MaxSpawnCount": 3,
    "TeamToSpawnInto": "Enemy",
    "ForceSpawnTemplateNames": ["NPC_TER_ShooterT0"],
    "EnemiesAmountNormal": [],
    "EnemiesAmountHard": ["NPC_TER_ShooterT0"],
    "NightmareMissionDifficultyTemplateNames": []
  }
}
```

8.3. EnemyStatsModifiers: difficulty settings

File `Configs/EnemyStatsModifiers.json`. Five numbers: how much the difficulty settings strengthen enemies.

Field	In the game	Meaning
`StrongDaamatPowerUp`	10	Daamat power bonus with the "strong Daamat" setting
`ResilientEnemiesPowerUp`	15	Enemy health bonus with the "resilient enemies" setting, in percent
`AccurateEnemiesPowerUp`	10	Enemy accuracy bonus with the "accurate enemies" setting
`NightmareMissionEnemyHealthBuff`	15	Health bonus on `Nightmare` difficulty missions, in percent
`NightmareMissionEnemyAccuracyBuff`	5	Accuracy bonus on such missions

```
{
  "ResilientEnemiesPowerUp": 30
}
```

9. Random events

9.1. RandomEventsSettings

File `Configs/RandomEventsSettings.json`. `MinDays` and `MaxDays` are the range of days between events in the hub (10–14 in the game).

9.2. RandomEvents

File `Configs/RandomEvents.json`. An array of events merged by the `Id` field (see 1.3): a mod brings only its own events and the changed fields of existing ones. The game has 54 events.

Field	Type	Meaning
`Id`	string, key	Event identifier
`ImageId`	string	Event image
`TitleKey`, `DescriptionKey`	text key	Title and text
`IsUnique`	yes/no	The event happens once per campaign
`ParticipantSlots`	array	Participants. Element: `RoleKey` is the role name, `Selector` is how the mercenary is chosen. Replaced entirely
`Conditions`	array	Conditions: the event is possible only if all are met. Replaced as a whole
`Options`	array	Player options. Replaced as a whole

Options element

Field	Meaning
`ButtonTextKey`	Button text
`OutcomeTextKey`	Outcome text
`OutcomeImageId`	Outcome image, may be left empty
`Consequences`	Consequences of the choice

Selectors, conditions and consequences are objects with a `"$type"` field. The value is written as `"<full type name>, Code"`. Every consequence has a `ShouldShowLog` field: whether to show it in the results.

Participant selectors: `Game.HubSystem.RandomEventsSystem.Selectors.<Name>`

Name	Fields	What it does
`SpecificCharacterSelector`	`CharacterId`	A specific named mercenary
`RandomUniqueSelector`	`Gender`, `UniqueMode`, `AllowedStatuses`	A random mercenary with filters

`InheritParticipantSelector`	`ParentEventId`, `ParentRoleKey`, `AllowedStatuses`	The same mercenary who took part in another event

Conditions: `Game.HubSystem.RandomEventsSystem.Conditions.<Name>`

Name	Fields	When met
`RosterCountCondition`	`Amount`, `RequiredGender`, `UniqueMode`, `AllowedStatuses`	The squad has this many suitable mercenaries
`SpecificCharacterAvailableCondition`	`CharacterId`, `AllowedStatuses`	A named mercenary is in the squad and in the required state
`InheritParticipantAvailableCondition`	`ParentEventId`, `ParentRoleKey`, `AllowedStatuses`	A participant of a past event is available
`HasResourceCondition`	`ResourceID` (`"money"`, `"intel"`), `Amount`	The resource is at least the given amount
`FacilityBuiltCondition`	`FacilityId`	A compartment is built, for example `"saloon"`
`FactionUnlockedCondition`	`Faction`	A faction is unlocked
`EnemyStrengthCondition`	`TargetStrength`, `Operator`	Daamat power is compared with a number
`PreviousEventPlayedCondition`	`EventId`	The event has already happened
`PreviousEventChoiceCondition`	`EventId`, `RequiredOptionIndex`	The option with this number (from zero) was chosen in a past event
`FreeSpaceInBarracksCondition`	—	The barracks have room

Consequences: `Game.HubSystem.RandomEventsSystem.Consequence.<Name>`

Name	Fields	What it does

`ModifyResourceConsequence`	`ResourceId`, `Amount`	Changes money or intel; a negative number is a deduction
`AddExperienceConsequence`	`TargetRoleKey`, `Amount`	Gives the participant experience
`SetStatusConsequence`	`TargetRoleKey`, `NewStatus`, `duration`	Puts the participant into a state for a number of days
`CharacterBuffConsequence`	`TargetRoleKey`, `BuffId`, `DurationDays`	A temporary bonus from `CharacterMetaBuffs` (3.5) for the participant
`GlobalBuffConsequence`	`TargetGender`, `BuffId`, `DurationDays`	The same bonus for all mercenaries of the given gender
`FacilityBuffConsequence`	`FacilityId`, `BuffID`, `ModifierPercent`, `DurationDays`	Temporarily strengthens or weakens a compartment
`AddTrophyConsequence`	`TrophyId`, `Amount`	Grants trophies
`ChangeReputationWithFaction`	`Factions`, `Amount`	Changes reputation with a faction
`RecruitCharacterConsequence`	`ArchetypeId`, `Gender`	Adds a new mercenary to the squad
`DismissCharacterConsequence`	`TargetRoleKey`	Removes the participant from the squad
`TriggerEventConsequence`	`NextEventId`	Starts another event

Numeric values in events. In events, lists are written as numbers, not names.

Field	Values
`Gender`, `RequiredGender`, `TargetGender`	0 — male, 1 — female, 2 — any
`UniqueMode`	0 — any, 1 — named only, 2 — unnamed only
`AllowedStatuses`, `NewStatus`	0 `None`, 1 `Ready`, 2 `Relax`, 3 `InjuredEasy`, 4 `InjuredMedium`, 5 `InjuredHard`, 6 `OnEvent`, 7 `OnMission`, 8 `Captured`, 9 `Training`, 10 `Dead`
`ResourceId` in a consequence	0 — money, 1 — intel
`Faction`, `Factions`	0 `Shield`, 1 `TsaiShen`, 2 `Front`

`FacilityId` in a consequence	0 `logistic`, 1 `saloon`, 2 `hospital`
`TrophyId`	0 `valuables`, 1 `equipment`, 2 `electronics`, 3 `jewelry`, 4 `antiquity`, 5 `alcohol`, 6 `sigares`, 7 `drugs`, 8 `diamonds`, 9 `cans`
`Operator`	0 `>=`, 1 `<=`, 2 `=`, 3 `>`, 4 `<`

Example: make an existing event repeatable and add your own.

```
[
  { "Id": "Armwrestling", "IsUnique": false },
  {
    "Id": "ModTestEvent",
    "ImageId": "Armwrestling",
    "TitleKey": "event/ModTestEvent/title",
    "DescriptionKey": "event/ModTestEvent/text",
    "IsUnique": false,
    "ParticipantSlots": [
      {
        "RoleKey": "Lucky",
        "Selector": {
          "$type":
"Game.HubSystem.RandomEventsSystem.Selectors.RandomUniqueSelector, Code",
          "Gender": 2,
          "UniqueMode": 2,
          "AllowedStatuses": [1, 2]
        }
      }
    ],
    "Conditions": [
      {
        "$type":
"Game.HubSystem.RandomEventsSystem.Conditions.RosterCountCondition, Code",
        "Amount": 1,
        "RequiredGender": 2,
        "AllowedStatuses": [1, 2],
        "UniqueMode": 2
      }
    ],
    "Options": [
      {
        "ButtonTextKey": "event/ModTestEvent/choice_1/title",
        "OutcomeTextKey": "event/ModTestEvent/choice_1/text",
        "OutcomeImageId": "",
        "Consequences": [
          {
            "$type":
"Game.HubSystem.RandomEventsSystem.Consequence.ModifyResourceConsequence, Code",
            "ResourceId": 0,
            "Amount": 12345,
            "ShouldShowLog": true
          }
        ]
      }
    ]
  }
```

```
]
```

The texts of a new event are added to the mod's `Localizations/English` under the keys from `TitleKey`, `DescriptionKey`, `ButtonTextKey`, `OutcomeTextKey`.

10. Mission scenarios

10.1. How a scenario file is organized

Folder `Configs/Scenarios`; the game has 62 files. One file is the scenario of one mission: what happens on the map, the objectives, when you win and lose. A combat map refers to the scenario with the `missionId` field (section 11): a map with `"missionId": "Generic_Rescue"` runs `Scenarios/Generic_Rescue.json`. Scenarios named `Generic_*` are shared by many maps; `S01_*`–`S15_*` and `DLC*` are story scenarios.

A mod can edit an existing scenario or bring a whole new file.

Field	Type	Meaning
`missionVersion`	number	Scenario version. ⚠ An in-combat save made with a different version restarts the mission
`scenariosData`	object	Scenario blocks: the key is the block identifier

A scenario block is an independent chain of steps. All blocks of a mission run in parallel: each waits for its start conditions, then performs its steps in order.

Field	Type	Meaning
`scenarioId`	string	Identifier, same as the key
`scenarioType`	list	`Completable` everywhere in the game
`startConditionsData`	array	Start conditions: the block starts when all are met
`stepsData`	array	Steps in order. Replaced as a whole
`enabled`	yes/no	A disabled block does not start
`ignoreStartConditions`	yes/no	Start at once without checking the conditions
`highPriority`	yes/no	⚠ The block runs before the others

Because the array is replaced entirely, to change one step of an existing block the patch must carry the block's whole `stepsData`. A new block is added simply as a new key.

10.2. Start conditions

A `startConditionsData` element: the `type` field and a data block under its own name.

Type	Data	When met
`ScenarioCompletedCondition`	`scenarioCompletedCondition`: `scenarioIds`	All listed blocks are completed
`ScenarioNotCompletedCondition`	`scenarioNotCompletedCondition`: `scenarioIds`	None of the listed blocks is completed
`ScenarioIsNotRunningCondition`	`scenarioIsNotRunningCondition`: `scenarioIds`	The listed blocks are not running now
`AllSpartansDeadCondition`	`allSpartansDeadScenarioCondition`: an empty object	All of the player's mercenaries are out
`AllSpartansEvacuatedCondition`	`allSpartansEvacuatedScenarioCondition`: an empty object	All of the player's mercenaries are evacuated
`LoseSpartanCondition`	—	A mercenary is lost
`AllAlliesDeadCondition`	—	All allies are dead

10.3. Steps

A `stepsData` element: the `stepType` field and a parameter block. **Each step has its own name for the parameter block**, and it can't be derived from the type by a rule: take it from the table in appendix B.

```
{ "stepType": "DelayStep", "delayStep": { "delay": 1.0 } }
```

Steps are of two kinds: an action (done, move on) and a wait (the chain stands until an event happens). Steps find objects on the map by tags: a tag is set on a map object or on a unit when it spawns (11.2). The targetTag parameter is the step's target tag. A detailed description of every step and all its parameters is in appendix B; it is taken from the developers' documentation in the game code.

Scenario flow

Step	Parameters	What it does

`DelayStep`	`delay`	Pause in seconds
`WaitNumberOfTurnsStep`	`turns`, `showCounter`, `localizationKey`, `turnsUpdateIcon`	Waits for the given number of turns; can show a counter
`WaitForFactionTurnStep`	`factionIdx`	Waits for a side's turn to start: 1 is the player, 2 allies, 3 enemies, 4 civilians
`ForceCompleteScenarioStep`	`idToComplete`, `isSuccess`	Forces another block to complete
`CompleteMissionStep`	—	Victory
`FailMissionStep`	—	Defeat
`AutoSaveStep`	—	Autosave
`AddChoiceStep`	`choices`, `isVideoCall`	Records story decisions; goods, contracts and rewards check against them
`CheckPlayerChoicesStep`	`playerHaveChoices`, `playerDoesntHaveChoices`	Checks that the player has some decisions and lacks others
`UpdateCounterStep`	`key`, `value`, `isSet`	Changes a named counter: `isSet` writes the value, otherwise adds to it
`WaitCounter`	`counterKey1`, `counterKey2`	⚠ Waits until the first counter equals the second

Combat wait conditions

Step	Parameters	What it waits for
`UnitInSight`	`targetTag`	The player has seen a unit with the tag
`NoAliveEntitiesWithTagStep`	`targetTag`, `ignoreStunned`, `onlyActualDeath`	No living units with the tag are left
`TriggerInRangeStep`	`targetTag`	A trigger object with the tag ended up on a cell

		occupied by the target
`UnitWithTagHpCheckStep`	`targetTag`, `relation`, `targetHp`, `checkAllTargets`	A unit's health is compared with a number: "==", "!=", ">", "<", ">=", "<="
`IsEntityInRadiusStep`	`anchorTag`, `targetTag`, `radius`	A unit is within a radius of an object
`WaitForFirstKillStep`	`faction`	The first kill of a side's unit
`CheckAbilityUsedAtTargetStep`	`abilityIdent`, `targetTag`	An ability was used on the target
`CheckObjectiveCompleteStep`	`objectiveTag`, `objectiveAmount`	An action was performed on the target object (take documents, open a stash)
`AliveCiviliansLeftStep`	`targetTag`, `percentToComplete`	The share of living civilians fell to the threshold
`RescueUnitWithTagStep`	`unitsTag`, `percentToComplete`, `uiTaskKey`, `mainUi`	The required share of units is rescued; shows the objective
`CaptureAndEvacuateMissionStep`	`targetTag`, text and counter keys, `forceCompleteScenario`	Composite objective «stun and carry out»
`StartLocationCheckStep` and `WaitForLocationCheckStep`	`id`, `unitsTag`, `triggerTag`, `targetUnitsNumber`, `turnsToWait`	Start and wait for the check «this many mercenaries stand in the zone»
`WaitForCharacterWithTagEvacuationStep`	`targetTag`	A unit is evacuated

Units and map

Step	Parameters	What it does
`PlaySpawnScenarioStep`	`scenarioId`,	Creates units from

	`spawnImmediately`, `lockScampBeforeSpawn`, `doNotFocusCamera`	a set in the map's `spawnScenario` (11.2)
`SwitchObjectsByTagStep`	`targetTag`, `setEnabled`	Enables or disables objects with the tag
`ForceVisibleEntity`	`entityTag`, `enable`	Makes units visible through the fog of war
`MoveCharacterWithTagToPointStep`	`targetTag`, `positionX`, `positionY`, `floor`, `isTeleport`, `waitForCompletion`	Walks or teleports a unit to a cell
`KillCharacterByTagStep`	`targetTag`	Kills a unit
`ReviveCharactersByTagStep`	`targetTag`, `hp`	Revives units with the given health
`AddBuffOnTargetByTagStep`	`targetTag`, `buffId`, `removeBuff`	Applies or removes a buff from `Buffs`
`RemoveAllStatusEffectsByTagStep`	`targetTag`	Removes all status effects
`SetCharacterActionPointStep`	`targetTag`, `commonPoints`, `attackPoints`, `movePoints`	Sets a unit's action points
`SetUnitFactionStep`	`targetTag`, `faction`	Changes a unit's side
`SetCharacterToPlayerByTagStep`	`targetTag`, `setToPlayer`	Puts a unit under the player's control
`SetTagStep`	`targetTag`, `tagToAddOrRemove`, `isTagAdded`	Adds or removes a tag
`MakeCharacterUseAbilityAtTargetStep`	`unitTag`, `abilityIdent`, `targetTag`	Makes a unit use an ability
`SetAIPriorityTargetStep`	`priorityTargetTag`, `workerAmount`, `taskRadius`	Assigns a priority target to the AI
`AIThinkSwitchStep`,	`setEnabled`	Disables the AI;

`ForceCompleteAITurn`		makes the AI end its turn immediately
`ScampSwitchStep`	`setEnabled`	Enables or disables enemies scattering to cover when detected
`LockDoorStep`, `OpenDoor`	`doorTag`, `setLocked` or `isOpen`, `isInstant`	Locks and opens doors
`EnableLockCellStep`, `BlockAllCellsButOneStep`, `UpdateCoverCellState`, `RestrictOverCell`	cell coordinates `x`, `y`, `i`	Block cells, cover and observation, mostly for the tutorial
`PlantBombStep`, `ExplodeBombStep`	radius, delay, timer	Planting and detonating a bomb in sabotage missions
`SwitchFogStep`	`setEnabled`	Fog of war
`LockCallEvacuationStep`	`setLocked`	Forbids calling an evacuation

Interface, camera, sound

Step	Parameters	What it does
`ShowUIStep`	`localizationKey`, `main`, `isCounter`, `currentValue`, `maxVale`	Shows a mission objective; with `isCounter`, with a counter of two named values
`UpdateUIStatusStep`	`localizationKey`, `state`, `isSuccess`	Changes an objective's state: completed, failed
`RemoveUIStep`	`localizationKey`	Removes an objective
`ShowHintMarkStep`, `ShowMissionMarkerStep`	`targetTag`, `spritePath`, `yOffset`, `show`	Marker above an object
`HighlightCellStep`	`x`, `y`, `floor`, `enableHighlight`	Cell highlight
`ShowModalWindowStep`	`id`	A window from

		ModalWindowsConfigs
PlayDialogueStep	id, waitForCompletion, showSubtitles	Plays a dialogue from section 13
SwitchRandomReplicasStep	enable	Enables or disables random mercenary lines
PlaySoundStep, StopSoundStep, PlayStoryMusicStep	id or musicPair, musicEvent	Sound and music
PointCameraStep	targetTag, height, angle, transitionTime, waitForCompletion	Points the camera at an object
ReturnCameraToSelectedCharacterStep	—	Returns the camera to the selected mercenary
StartCutsceneCameraStep, ShowCinematicStep, WaitForCutsceneCompleteStep	cameraTag, cutsceneId	Cutscenes built into the map scene
FadeScreenStep	fadeTime, showFade	Screen fade
SwitchControlsStep	enabled	Disables player control
LockTurnTransitionStep	setLocked	Forbids ending the turn
HelicopterAnimationStep, ForceAnimationPlayStep, WaitAnimationEventStep	state, animation name	A character descending from a helicopter, object animations, waiting for an animation event

Hub steps. The _HubMission scenario drives the tutorial and the story at the base: SpawnMissionOnMap, WaitMissionStartStep, WaitBuildStep, WaitContract, WaitScreenStep, HubCustomUI, TutorialMarker, AddNewsStep, PlayVideoCall, SetFactionProfileStep, DelayMetaStep, WaitMetaDateTime, AddCharacter and others. Their parameters are in appendix B.

Example: a new block in the shared rescue scenario that shows a hint dialogue three turns after the start.

```
{
  "scenariosData": {
    "ModHint": {
      "scenarioId": "ModHint",
      "scenarioType": "Completable",
      "stepsData": [
        { "stepType": "WaitNumberOfTurnsStep", "waitNumberOfTurnsStep": {
```

```
"turns": 3, "showCounter": false } },
        { "stepType": "PlayDialogueStep", "playDialogueStep": { "id":
"Generic_mission_escape_landing_02", "waitForCompletion": false, "showSubtitles":
true } }
      ],
      "startConditionsData": [
        { "type": "ScenarioCompletedCondition", "scenarioCompletedCondition": {
"scenarioIds": ["Start"] } }
      ],
      "enabled": true,
      "ignoreStartConditions": false,
      "highPriority": false
    }
  }

}
```

Mod file: `Configs/Scenarios/Generic_Rescue.json`.

10.4. ScenarioServiceConfig and map counters

`Configs/ScenarioServiceConfig.json`: `hubMissionName` is the name of the hub scenario, `hubMissions` is the list of hub scenarios.

`Configs/CombatMapCounterTemplatesConfig.json`: turn counter presets (for example, until reinforcements) that depend on Daamat power. Object: the key is the preset name (`default`, `Preset0` …), the value is an array of rows `PowerMin`, `PowerMax`, `CounterValue` (number of turns). A map chooses a preset with the `counterId` field. The array is replaced entirely.

```
{
  "Preset0": [
    { "PowerMin": 0, "PowerMax": 100, "CounterValue": 12 }
  ]

}
```

11. Combat maps

11.1. What you can and can't do

Folder `Configs/CombatMaps`; the game has 640 files. A map file describes a finished game scene: geometry, cover and objects live in the scene itself, and the config lists them and sets the combat parameters. So a mod can't create a new location. You can:

- change the parameters of an existing map: start point, spawn types, scenario, counter, unit sets;
- bring a new map file that uses an existing map's scene (a different mission at the same location) and add it to the mission type's `mapIds` (7.1).

Field	Type	Meaning
`scenePath`	string	Game scene. Must not be changed

`mapSize`	`[x, y]`	Grid size in cells
`startPoint`	`[x, y, floor]`	Squad landing cell
`startPointSize`	`[x, y]`	Landing zone size
`minBorders`, `maxBorders`	`{x, y}`	Playable area bounds
`missionId`	string	Scenario from `Configs/Scenarios` (section 10)
`SpawnIdent`	string array	What appears at the map's spawn points, in point order: a spawn type (`SpawnType` from `AreaSpawnEnemyPowerConfig`, 8.1) or a specific character template. Replaced entirely
`spawnScenario`	array	Named unit sets for the `PlaySpawnScenarioStep` step. Replaced entirely
`counterId`	string	Turn counter preset from `CombatMapCounterTemplatesConfig`
`mapVersion`	number	Map version. An in-combat save with a different version restarts the mission
`locationAmbient`	string	Location ambient sound
`briefPrefabKey`	string	Briefing appearance
`mapEntities`	array	Scene objects: `entityId`, `path` (path to the object in the scene), `tags`, `teamTag`, `faction`, `rotation`. Replaced entirely and must match the scene exactly: don't touch it

11.2. Unit sets: spawnScenario

Field	Type	Meaning
`scenarioId`	string	Set name; the `PlaySpawnScenarioStep` step calls the set by it
`SpawnScenarioStep`	array	Unit groups
`…points`	array `[x, y, floor]`	Spawn cells
`…characters`	string array	Character templates from `CharacterTemplates` (3.2)
`…tags`	string array	Tags by which scenario steps find the units
`…faction`	list	Side: `Enemy`, `Sparta`, `Allies`, `Civilian`, `None`

…alertState	number	Alert state: 0 — calm, 1 — alerted, 4 — in combat
…spawnFromMeta	yes/no	⚠ Take units from the player's squad

Example: on the S03_OfficerCapture map, strengthen the last two spawn points and change the counter preset. spawnScenario and mapEntities are not mentioned in the patch and stay as in the game. Mod file: Configs/CombatMaps/S03_OfficerCapture.json.

```
{
  "SpawnIdent": ["NPC_TER_Officer", "reinforcement", "reinforcement",
"reinforcement", "reinforcement", "reinforcement", "support", "big", "sniper"],
  "counterId": "Preset1"
}
```

The number of SpawnIdent elements must match the number of spawn points on the map.

12. Character appearance

12.1. ClothingConfig: clothing sets

Folder Configs/ClothingConfig; the game has 191 files. One file is one set: which parts the character model is assembled from. The file name is the set identifier. A mod can change sets and bring new files.

Field	Type	Meaning
clothingParts	object	Slot → array of part identifiers
selectedDecals	object	Tattoos and scars: slot number → decal number
selectedDecalVariants	object	Decal variant: slot number → variant number
selectedPortraitIdent	string	Portrait

clothingParts slots: HeadSkin (head), Hair, Hat, Mask, HeadBottom, HeadMisc1, HeadMisc2, UpperBody, LowerBody, UpperBodyPatch1, BodyMisc1, BodyMisc2, and also UpperBodyG9, LowerBodyG9: clothing under G9 armor. The slots _ArmorTier1 … _ArmorTier30 set how armor looks on this character: the number is the armor's maleArmorVisualTier or femaleArmorVisualTier value (5.2).

A part identifier is a path like male/body/up/tshirt_02_427. Only parts that exist in the game can be used; they are listed in CharacterRedactorItemsDatabase.json. A slot's array is replaced entirely.

```
{
  "clothingParts": {
    "UpperBody": ["male/body/up/shirt_01_003"],
    "Hat": []
  }
```

```
}
```

12.2. Modules/UpdateMeshConfig: linking appearance to a character

Object: the key is the preset identifier set in the character template in the `UpdateMesh` module (3.2).

Field	Meaning
`defaultClothingIdent`	Clothing set from 12.1 (file name without extension)
`meshKeys`	Ready-made models when the character is not assembled from parts (enemies)
`masterRootPrefab`, `interactiveMasterRootPrefab`, `briefRootPrefab`	Model base for combat, the base and the briefing: male or female
`isNonHuman`	Not a human: drone, mine, item

To change a character's clothes: create a set file in `ClothingConfig`, add a preset to `UpdateMeshConfig` with your `defaultClothingIdent`, and put the preset into the character template.

12.3. ClothingConfig/ClothesTexturesConfigs: clothes textures and the editor

Folder `Configs/ClothingConfig/ClothesTexturesConfigs`. One file is one clothing item for the character editor: the model, the shaders of its materials and the color variants. Each «tier / color» pair in the file is a separate item in the character editor. A mod can change the game's configs and bring new ones.

The file name is `{item}_{category}_{gender}`, for example `party_Hat_Male`. The category and gender in the name must match the `itemCategory` and `gender` fields: the game skips a config with a different ending.

Field	Type	Meaning
`gender`	string	Character gender: `Male` or `Female`
`itemCategory`	string	Item category: `Hat`, `UpperBody`, `Armor`, `ArmorPatches`, `ClothesPatches`, etc.
`sourceFbx`	string	Path to the model
`spawnFromAddressables`, `prefabAddressable`	boolean, string	Take a ready-made game prefab from Addressables by address instead of building it from

		`sourceFbx`
`shaders`	array	Shader of each material, one element per material: `HDRP/Lit`, `HDRP/Hair`, `Shader Graphs/DigitalHuman_Hair`. The game substitutes `HDRP/Lit` for an unknown one
`baseTextures`	array	Default textures and shader properties, one element per material
`materials`	array	One element per material: tier → color → textures. The tier is written with two digits (`00`), the color with three (`002`)
`paintableMesh`	array	One element per material: body parts on which tattoos can be painted over it (`Head`, `UpperBody`, `Back`, `Hips`, `LeftArm` …)
`canBeUsedOnlyWith`	array	Parents: names of the clothing configs this item can be worn together with

A `baseTextures` element and a variant in `materials` have the same structure: `baseTexture`, `maskTexture`, `normalTexture` and `properties`, the remaining shader properties by type (`textures`, `floats`, `colors`, `vectors`, `diffusionProfiles`). `baseTextures` holds all shader properties, a variant only the overridden ones: the game takes the default properties and applies the variant's properties on top.

```
{
  "gender": "Male",
  "itemCategory": "Hat",
  "sourceFbx": "my_mod/Assets/MyHat/hat.fbx",
  "shaders": ["HDRP/Lit"],
  "baseTextures": [
    { "baseTexture": "my_mod/Assets/MyHat/hat_base.png", "maskTexture": null,
      "normalTexture": "my_mod/Assets/MyHat/hat_normal.png" }
  ],
  "materials": [
    { "00": { "001": { "baseTexture": "my_mod/Assets/MyHat/hat_red.png",
                       "properties": { "floats": { "_Smoothness": 0.3 } } } } }
  ]
}
```

File paths. A game asset is given by its path from the Unity project root: `Assets/ProjectAssets/…`. A mod file is given by its path from the `Mods` folder, starting with the mod folder name: `my_mod/Assets/MyHat/hat.fbx`. This way you can refer to a file of any mod in the `Mods` folder, not only your own; the file can be anywhere inside the mod folder. The game won't find a path like `Assets/StreamingAssets/Mods/…`. A mod model is a binary FBX 7.x, OBJ, glTF 2.0 or GLB skinned to the game skeleton; mod textures are png or jpg. A third-party model is first transferred to the game skeleton with the `WarhoundsModEditor.exe --prepare-fbx` command.

Parents. Patches, armor attachments and hair under a hat without `canBeUsedOnlyWith` don't make it into the game. Armor patches and attachments (`ArmorPatches`, `ArmorMisc`) attach only to armor (`Armor`), clothes patches (`ClothesPatches`) only to upper-body clothing (`UpperBody`).

The editor in Mod Editor

Button and menu names are those of the English version of the editor. The configs are in the tree on the left, in the Clothes textures section. A clothing config has a single entry, so the window is laid out differently from other configs: at the top a short field table (gender, category, Addressables), below it a materials panel across the full width of the window.

Panel element	What it does
Model	The `sourceFbx` path. Next to it is the Add localization button
Material, Shader	Choice of the material slot and its shader. Add slot creates a slot with the previous slot's shader and properties; Remove slot deletes the slot together with its variants, shader and body parts
Paintable body parts	The slot's body parts (`paintableMesh`): Add part, Remove part
Only together with	Parents (`canBeUsedOnlyWith`): a drop-down list of clothing configs of the same gender and a suitable category; the name can be typed
Slot variants	A «tier / color» table with base, mask and normal texture paths. The first row is the default textures (`baseTextures`)
Shader properties	To the right of the variants table: the shader properties of the selected row. The border between them can be dragged

Variants. Add creates a variant from the default row, Clone copies the selected variant together with its shader properties, Remove deletes it. Tier and color are entered separated by a space («00 025») and are edited right in the table. Green rows were added by the mod, yellow ones changed. To the left of a path is a texture thumbnail; a gray placeholder means the texture is not set or not found. Paths are suggested from the game's textures and from mod folders.

Files from Explorer. A model or texture can be dropped with the mouse onto the model field, onto a texture thumbnail or path field, and onto a texture row in the shader properties. The editor writes the correct path itself: from the `Mods` folder for a mod file, from the project root for a Unity project file. A file located elsewhere is rejected with an explanation.

A red path means there is no such file either in the mod folders or among the game assets, or the path leads into the project's `StreamingAssets` through a mod folder. The reason is in the tooltip. In a built game the assets are packed and a game asset path can't be checked, so it is not red.

Shader properties. The default row shows all shader properties with readable labels (Alpha

Clip Threshold rather than the internal name). Numbers are edited with a slider and the field next to it; a number can be dragged left and right by its label (Shift is faster, Ctrl is finer). Checkboxes and lists are edited with their own controls, a color with a color picker window with RGBA fields: while you pick a color it is already visible in the 3D preview, and Cancel restores the previous one. A gray value is the shader's default value; only changed values go into the config; «×» restores the default. Properties hidden by a shader condition (the threshold when Alpha Clipping is off) are not shown. The mouse wheel over a slider or list only scrolls the panel.

A «tier / color» variant shows only the overridden properties: a new one is added from the list at the bottom, «×» removes the override. The JSON… button under the list opens the same properties as text, which makes them easy to copy from other configs.

3D preview (the 3D preview... button, the View menu or F6). Shows the model from `sourceFbx` with the config's materials: the default values, or, if a variant is selected, with its override. The window follows the selected config and redraws on every edit. The left mouse button rotates the model, the middle one pans, the wheel zooms, Reset view resets the camera. The slider changes the lighting; Highlight the selected slot colors the slot selected in the panel. The preview is approximate: no shadows, SSS or post-processing, in the bind pose. The preview reads more formats than the game (tga, psd), so rely on the F5 validation.

Item name. Add localization adds the item to `CharacterRedactorItemsLocalizationAndSprites` with the text key `redactorItem/{gender}/{category}/{item}` (for example, `redactorItem/male/Hat/myHat`), creates this key in the mod's texts and opens the texts at that line. All that is left is to type the name.

Config name. When `gender` or `itemCategory` is changed in the field table, the config is renamed immediately (`party_Hat_Male` → `party_Armor_Male`). If the new name is taken, the field reverts to its previous value. A game config is not renamed; a warning appears in the status bar instead.

Duplicate, Rename, Delete. Right-click a config in the tree:

- Duplicate: a copy with the current contents, including the mod's edits, created right away as a file in the mod folder. This is how a new mod item is made from a game item. The name is entered without the `_{category}_{gender}` suffix; the editor appends it.
- Rename… (F2): only for a config that exists only in the mod.
- Delete… (Del): deletes the file of a config created in the mod, after confirmation. A game config can't be deleted.

References to the old name in other configs' `canBeUsedOnlyWith` are not updated on rename; the validation will find them.

Validation (F5) finds: a file name that doesn't match `itemCategory` and `gender`; `shaders`, `baseTextures` and `paintableMesh` lengths that don't match the number of materials; an unknown shader or body part; tier and color keys not in the `00` / `002` format; a config without a single variant (the item won't appear in the game); a patch or attachment without parents; non-existent models and textures; paths to mod files given from the Unity project; mod textures that are not png or jpg; a mod model in an unsupported format, without a skin, or with bones that don't match the game skeleton.

Limitations:

- When applying a patch, the game replaces the `materials` and `baseTextures` arrays

entirely: a clothing config patch contains the whole `materials`, and two mods editing the same config conflict.
- The order of material slots must match the order of materials in the model.
- The editor takes the list of shader properties from its own `Resources/ShaderProperties.json` file and doesn't know other shaders.

13. Dialogues, lines and news

13.1. General

All these configs hold references, not texts: `text` is a localization key, `sound` is a voice-over identifier, `character` is who speaks (the character's `metaInfoIdent` value). To change the text itself you don't need to touch the config; the localization file is enough. The config is edited when you need to change the set of lines. A mod can't add new voice-overs: give an existing sound or leave the field empty.

The game reads the processed copies from the English subfolder, not the files in the root of the folder: they have `duration` filled in. The patch must go at the same path.

What	File the game reads
Mission dialogues	`Dialogues/English/DialoguesProcessed.json`
Random mercenary lines	`Dialogues/English/ReplicasProcessed.json`
Briefings for regular missions	`BriefDebrief/English/BriefReplicaConfigProcessed.json`
Debriefings for regular missions	`BriefDebrief/English/DebriefReplicaConfigProcessed.json`
Briefings and debriefings for story missions	`BriefDebrief/English/StoryBriefDebriefReplicaConfigProcessed.json`
Video calls	`VideoCall/English/VideoCallConfigProcessed.json`
Mercenary conversations in combat	`GameTalk/English/GameTalkConfigProcessed.json`
Conversations in the barracks	`BarracksDialogue/English/BarracksDialogueProcessed.json`
Comments on map objects	`ObjectClick/English/ObjectClickModuleProcessed.json`

The folder is called `English` for any game language: the text language is determined by

localization.

13.2. A line

Field	Type	Meaning
`character`	string	Who speaks
`sound`	string	Voice-over
`text`	text key	Subtitle
`duration`	number	Duration in seconds; `-1` means by the sound length
`replicaType`	list	`RandomReplica`, `StoryDialogue`, `AdditionalDialogues`, `BarracksDialogue`, `GameTalkDialogue`
`isRandom`, `replicas`	yes/no, array	Instead of one line — a random one from the nested list
`ignoreMetaInfo`	yes/no	⚠ Do not insert the speaker's name and portrait

13.3. File formats

File	Format
Mission dialogues	Object: dialogue identifier → array of lines. The identifier is called by the `PlayDialogueStep` step
Random lines	Object: character → event → block: `chanceToPlay` (chance from 0 to 1), `priority`, `replicaCollection` (array of lines to choose from). Events: `Select`, `Move`, `Attack`, `Enemy_kill`, `Miss`, `Wounded`, `Reload`, `Overwatch`, `Grenade`, `Enemy_detected`, `Friend_killed` and others, about 70 in total, including separate ones for abilities (`Ab_sprint`, `Ab_headshot`)
Briefings and debriefings for regular missions	An object with arrays of variants. A variant is chosen by `missionType`, `missionSource`, `enemy`, `goalsCompleted`, `missionCase`; `characterReplicas` maps a role → list of line chains
Briefings and debriefings for story missions	`Briefs` and `Debriefs` are arrays; element: `missionId`, `timelines` (main text), `commentariesOnTimeline` (mercenary comments by personality type), `uniqCommentaries` (comments of named mercenaries), `additionalCharactersInBrief`
Video calls	`videocalls` is an array: `trigger` (what calls it), `character`, `replicas`
Combat	`Chance` is the conversation chance in percent; `GameTalks` is an object: pair of characters → `characters` and `talks` (an array of dialogues; a

conversations	dialogue is an array of lines)
Conversations in the barracks	`Dialogues` is an object: identifier → character → array of lines (`Character`, `Text`, `Sound`, `Duration`)
Object comments	Object: object identifier → `hoverName` (hover caption) and `replicaCollection`

Line arrays are replaced entirely.

Example: Simba comments on mercenary selection less often. File `Configs/Dialogues/English/ReplicasProcessed.json`:

```
{
  "simbaCharacter": {
    "Select": { "chanceToPlay": 0.2 }
  }
}
```

13.4. News/NewsConfig: news

Field	Meaning
`RandomNews.period`	Every how many days a random news item is published
`RandomNews.news`	Object: campaign stage (`act_1`, `act_2` …) → array of news items
`CustomNews`	Array of story news; shown by the `AddNewsStep` scenario step

A news item: `newsId`, `newsTitle`, `lead`, `newsText` (text keys), `newsImg` (sprite key). The arrays are replaced entirely.

13.5. videoSubtitles: video subtitles

Folder `Configs/videoSubtitles`, one file per video. `videoSubtitles` is an array of rows: `phraseKey` (text key), `startTime` and `endTime` in seconds from the start of the video.

14. Other configs

14.1. PlayerDefault: the campaign starting state

File `Configs/PlayerDefault.json`. This is the template for a new game's save. The most useful fields:

Field	Meaning
`currency`	Starting resources: `money`, `intel`, `crew`, `characters` (mercenary slots)

`DaamatPower`	Starting Daamat power
`characters`	Starting squad: identifier → character in the `CharactersShop.uniqueCharacters` format (3.4)
`storedEquipment`	Starting storage content: identifier → quantity
`availableEquipmentForBuy`	What is on sale from the start
`Factions`	Starting reputation with factions
`PlayerCompartmentData`	Compartments built from the start
`prologueMissionsIds`	Prologue missions in order
`newsAct`	Starting news stage

The other fields are service ones. Changes apply only to a new campaign.

```
{
  "currency": { "money": 150000 },
  "storedEquipment": { "FragGrenade": 4 }
}
```

14.2. TrophiesConfig: trophies

File `Configs/TrophiesConfig.json`. Object: the key is the trophy identifier (`equipment`, `electronics`, `jewelry`, `alcohol` …). The fields are as for an item (5.2): `ident`, `tier`, `itemName`, `itemDescription`, `itemIcon`, `iconSmall`, `sellPrice`, `itemType` (`Trophy`), `showInUI`, `stackableInStorage`. The main field is `sellPrice`, the selling price.

```
{
  "equipment": { "sellPrice": 400 }
}
```

14.3. Small configs

File	Fields	Meaning
`EvacuationSettings`	`evacuationZoneCallRechargeDuration`	Evacuation call cooldown in turns
`DeployableUnitsTemplates`	object: identifier → array of modules `configType` + `moduleIdent`	What deployable objects consist of: mines, bombs, smoke.

		An ability's `deployableId` (2.1) refers to the identifier
`UniqueCharactersBio`	`bioTemplatesByMetaInfoIdent`, `bioTemplates`	Biography template for a character and the template content: which text keys the biography is built from
`Modules/DoorModuleConfig`	`doorState` (`Closed`, `Open`), `isLocked`	Door presets
`Modules/DestructionModuleConfig`	`destructionHitPoints`, `destroyBehaviour`	Durability of destructible objects and whether the object disappears after destruction
`Modules/DestructionExplosionModuleConfig`	`damage` (like `damageAbilitySettings` in 2.1), `explosionSetting` (`explosionForm`, `explosionRadius`)	Explosion of barrels, gas cylinders, cars
`Modules/MapObjectiveGenericAbilityModuleConfig`	`objectiveID`, `title`, `description`, `icon`, `abilityAnimationTrigger`, `hideWeapon`	Actions with objective objects: free a hostage, take documents, plant a bomb
`Modules/ObjectClickModuleConfig`	`hoverName`, `replicaCollection`	Source of the object comments; the game reads the processed copy (13.1)
`Modules/WallConfig`, `FenceConfig`, `LadderConfig`	`updateCellPassability`, `updateCellHeight`	How walls, fences and

		ladders change cell passability and height. Tied to scene objects — better left alone

Appendix A. Value lists

A.1. Ability types: abilityType

`None`, `BaseAttack`, `Attack`, `RPGAttack`, `BaseOverwatch`, `Overwatch`, `TargetLocked`, `StealthOverwatch`, `RPGBaseOverwatch`, `AddBuff`, `AddStatusEffectOnStay`, `Reload`, `ReloadOrder`, `Open`, `Grenade`, `Timebomb`, `SuicideBeltExplosion`, `Healing`, `FirstAid`, `GetupAndFight`, `CreateDrone`, `TargetActive`, `TargetMy`, `ExplosionLast`, `ArmoryAround`, `AddActionPoint`, `BodyTransferUp`, `BodyTransferDown`, `HailOfBullets`, `DoubleTap`, `CanonShot`, `AOEAttack`, `AOEApplyStatusEffects`, `AOEStatusEffectsZoneAbility`, `MeleeKnife`, `Cutthroat`, `TogetherAbility`, `ProvenStimulantsAbility`, `NowOrNever`, `TacticalLessons`, `EntityGrenade`, `AddStatusEffect`, `AddStatusEffectForNextShot`, `FireFire`, `WillNotHide`, `QuickStart`, `TsarBomb`, `Taser`, `FieldSurgeon`, `MindbendingGrenade`, `Pyromania`, `ChangeOfPosition`, `Aura`, `AuraDieHard`, `AuraFreeCivilian`, `MapObjectiveGeneric`, `EvacuateSelf`, `EvacuateWithBody`, `EvacuateAll`, `HotBlood`, `Manoeuvre`, `ExtinguishingAbility`, `ExtinguishingSelf`, `ArmorRecovery`, `Regeneration`, `Roll`, `Provoke`, `ManicGaze`, `DynamicCamouflage`, `Scan`, `ThermalImager`, `Demining`, `Tie`, `Discharge`, `FatalDischarge`, `Kamikaze`, `UnderSelfGrenade`, `GiveGrenade`, `UnderSelfSmokeDrone`, `GetUp`, `PickLoot`, `ReconMaster`, `StatusEffectModifier`, `PierceTank`, `DestroyMine`.

The most versatile for mods: `AddBuff` (a passive or active ability that grants buffs from `abilitySettings.buffs`), `AddStatusEffect` (applies `statusEffects`), `Attack` (a shot with parameters from `weaponAbilitySettings`), `Grenade` (a throw with area damage), `Healing`.

A.2. Buff types: buffType

`Protection`, `CoverProtection`, `MovingTarget`, `Accuracy`, `AccuracyBonusForHeight`, `TearGasAccuracy`, `RepeatedAttackAccuracy`, `PoisonAccuracy`, `AccuracyForEntity`, `Mobility`, `Bleeding`, `Panic`, `Suppression`, `OverwatchProtection`, `OverwatchIgnoreDamage`, `OverwatchDamageIsScratch`, `OverwatchCharges`, `OverwatchShotCount`, `OverwatchInterruption`, `PersistantOverwatch`, `IronOverwatch`, `FovRadiusCharges`, `CriticalDamageChance`, `HitPoints`, `ArmorHitPoints`, `CommonSight`, `ScratchChanceIncreaseOnMiss`, `AccuracyIncreaseOnConsecutiveShots`, `DeathProtection`, `AbilityBuff`, `PullUpUnlock`, `Acrobatics`, `Exolegs`, `Tracer`, `ArmorFitting`, `AddActionPointsOnKill`, `RechargeAbilitiesOnKill`, `AddActionPointsOnDamage`, `Survival`, `AddActionPoints`, `AddActionPointsOnAllyKill`, `AddBuffInCover`, `BonusAttackOnMiss`, `AlwaysWinsInBrawling`, `DeathChanceOnZeroHp`, `GuaranteedSurvivalOnZeroHp`, `LuckyDodge`, `SpeedyTarget`, `Flickering`, `AddAttackPointOnTurnStart`, `AttackPoint`, `BoostAttackLastBulletDamage`, `AlwaysMaxDamage`, `AutoFirstAid`, `DamageResist`, `Stun`, `AddDebuffOnHit`, `AddDebuffOnAttack`, `ChangeAttackDamage`, `ChangeEntityAttackDamage`, `StatusEffectResistance`, `StatusEffectImmunity`, `Blindness`, `AvailableAirTransition`, `Flier`, `Crawler`, `ToughTarget`, `PreventAttackInterruption`,

`PreventMoveInterruption`, `CautiousFire`, `MarkEnemy`, `MarkedByTarget`, `MeleeFactor`, `SlashDamage`, `IncreaseTrophies`, `DynamicCamouflage`, `KillWithLastBullet`, `Restrain`, `AddActionPointsOnFirstIncapacitation`, `AirDefence`, `Electro`, `AddMeleeDamageMultiplierByStatusEffect`, `AddActionPointsOnAllyIncapacitationBuff`, `StreetsOfFuryBuff`, `AddStatusEffectsOnFlankingBuff`, `AddBuffsOnAttackSameTargetBuff`, `DisableAuraBuff`, `AddBuffAfterDamageType`, `ReplaceAbility`, `AddStatusEffectOnMiss`, `AddStatusEffectOnHit`, `NoMistake`, `RemoveTargetOverwatch`, `GuaranteedDeathOnZeroHp`.

Simple numeric buffs: `Accuracy`, `Protection` (defense), `CoverProtection` (defense in cover), `Mobility`, `CriticalDamageChance` (crit chance), `HitPoints` (health), `ArmorHitPoints` (armor), `MeleeFactor` (melee). The other types enable a special rule; use a game buff of the same type as a setup template.

A.3. Character classes: characterClass

Player mercenaries: `Newbie`, `Assault`, `Operator`, `Support`, `Sniper`, `Grenadier`.

Named mercenaries: `Assault_Shilo`, `Assault_Shaman`, `Assault_Brodyaga`, `Assault_Fay`, `Operator_Umka`, `Operator_Chapai`, `Operator_Strela`, `Support_Simba`, `Support_Diesel`, `Support_Ifrit`, `Sniper_Grif`, `Sniper_Blizzard`, `Sniper_Sunny`, `Sniper_Julia`, `Grenadier_Debut`, `Grenadier_Chuma`, `Grenadier_Snegovik`.

Drones: `DroneMedical`, `DroneAssault`, `DroneRecon`.

Enemies and others: `Bot`, `Daamat_Shooter`, `Daamat_Supressor`, `Daamat_Sniper`, `Daamat_Officer`, `Daamat_Machinegunner`, `Daamat_Grenadier`, `Daamat_Melee`, `Daamat_Suicider`, `Front_Soldier`, `Front_Demoman`, `Front_Officer`, `Armada_Pirate`, `Armada_Vigilante`, `Armada_RPG`, `TsaiShen_Operative`, `TsaiShen_Sniper`, `TsaiShen_Technician`, `TsaiShen_HunterDrone`, `Shield_Peacekeeper`, `Shield_Enforcer`, `Shield_Shieldbearer`, `Shield_DefenseDrone`, `G9_Legionary`, `Orphans_Bandit`, `Orphans_Bonecrusher`, `Orphans_Provoker`, `Wolves_Assault`, `Wolves_Specialist`, `Wolves_Machinegunner`, `Wolves_Grenadier`, `Wolves_Sniper`, `Turret`, `Vehicle`, `Civilian`, `VIP_Ally`, `VIP_Enemy`.

A mod can't add a new class.

A.4. Weapon classes: weaponClass

`None`, `Melee`, `Pistol`, `AutomaticRifle`, `Shotgun`, `SniperRifle`, `MachineGun`, `GrenadeLauncher`, `Taser`, `ClimbingHook`, `RocketLauncher`, `DualSubmachine`.

A.5. Mission types

Active: `Sweep`, `Elimination`, `Diversion`, `TargetCapture`, `AntiTerror`, `VipEscort`, `JointOperation`, `ObjectCapture`, `ObjectRescue`, `Rescue`.

Passive: `Spying`, `Theft`, `Deal`, `Search`, `Sabotage`, `ObjectSearch`, `ObjectGuard`, `Hunt`, `Ambush`, `Raid`, `Showdown`, `Evacuation`, `Research`, `CargoCapture`, `AgentMeeting`.

Special: `Investigation`, `BlackMarket`, `G9Search`.

Expansions: the `Sea` and `Ship` prefixes for the first, `SAR` for the second, `ANT` for the third (for example `SeaSweep`, `SARRescue`, `ANTSweep`).

A.6. Other

List	Values
Enemies and factions	`Daamat`, `Daamat_x_G9`, `G9`, `Armada`, `Orphans`, `Wolves`, `Shield`, `TsaiShen`, `Front`, `Unknown`, `None`
Damage types for resistances	`None`, `Kinetic`, `Impact`, `Fire`, `Electric`, `Explosion`
Equipment tier	`Tier1`, `Tier2`, `Tier3`
Inventory slots	`MainWeapon`, `Armor`, `EquipmentSlot`, `SpecialSlot`
Mission difficulty	`Normal`, `Hard`, `Nightmare`
Regions	`DefaultRegion`, `DLC1`, `DLC2`, `DLC3`

Appendix B. All scenario steps

The full list of step types used in the game's scenarios (140). The descriptions are taken from the developers' documentation in the game code (each step's data class) and are given almost verbatim. Under the step type is the name of its parameter block in JSON. The ⚠ sign marks places where the code has no description and it was inferred from the name.

Step and parameter block	What it does	Parameters
`AddBuffOnTargetByTagStep` / `addBuffOnTargetByTagStep`	Adds a buff to the target or removes it.	`targetTag`: tag of the target the buff applies to; `buffId`: buff id from the Buffs.json config; `removeBuff`: if false, the buff is added; if true, removed.
`AddCharacter` / `addCharacterStepData`	Adds a character to the team, bypassing the purchase procedure.	`characterId`: character ident from CharactersShop.json; No description in code: `addOnlyShop`
`AddChoiceStep` / `addChoiceStepData`	Adds a new entry to the player's choices.	`choices`: list of choices to record; `isVideoCall`: the choice starts a video call ⚠

`AddNewsStep` / `addNewsStepData`	Adds news for the player.	`newsIds`: news identifiers from News/NewsConfig.json (CustomNews)
`AIThinkSwitchStep` / `aIThinkSwitchStep`	Disables or enables the AI trees. Important: if you disabled them, don't forget to enable them again	No description in code: `setEnabled`
`AliveCiviliansLeftStep` / `aliveCiviliansLeftStepData`	Checks the number of living civilians on the map with the given tag.	`targetTag`: target tag; No description in code: `percentToComplete`
`AutoSaveStep` / `autoSaveStep`	Saves the game.	—
`BlockAllCellsButOneStep` / `blockAllCellsButOneStep`	Blocks/unblocks all cells for clicking except the one given by coordinates. You can't move to blocked cells or use an AOE ability on them.	`X,Y`: coordinates of the cell that won't be blocked; `unlock`: if true, all cells are unblocked.
`CameraRestrictZone` / `cameraRestrictZoneData`	Enables or disables a special camera restriction zone.	`enable`: enable or disable.; `x1, y1, x2, y2`: cells of two corners of the zone
`CaptureAndEvacuateMissionStep` / `captureAndEvacuateMissionStep`	Updates and waits for the counter of evacuated characters. Completes when the counter's current value equals the maximum.	`targetTag`: tag of the characters to be evacuated; in the full variable name it is appended to the prefix after the ":" sign; `showCounter`: whether to show the counter of evacuated targets; `main`: list the objective among the mission's main objectives or the optional ones; `<name>LocalizationKey`: localization key for the text of objective <name>; `<name>CurrentValuePrefix`: prefix of the variable (before the ":" sign) holding the counter's current value for objective <name>; `<name>MaxValuePrefix`: prefix of the variable

		(before the ":" sign) holding the counter's maximum value for objective <name>.; `<name>forceCompleteScenario`: the scenario that is forcibly completed when step <name> completes.; No description in code: `evacuationLocalizationKey`, `evacuationCurrentValuePrefix`, `evacuationMaxValuePrefix`, `stunLocalizationKey`, `stunCurrentValuePrefix`, `stunMaxValuePrefix`, `restrainLocalizationKey`, `restrainCurrentValuePrefix`, `restrainMaxValuePrefix`, `carryLocalizationKey`, `carryCurrentValuePrefix`, `carryMaxValuePrefix`, `callEvacuationLocalizationKey`, `callEvacuationCurrentValuePrefix`, `callEvacuationMaxValuePrefix`
`ChangeMoneyRewardStep` / `changeMoneyRewardStepData`	Changes the money reward on the current combat map. Below 1 the reward decreases (for example, 0.5: the player gets half of the initially set reward), above 1 it increases	No description in code: `changeDelta`
`CharacterLookAtTargetStep` / `characterLookAtTargetStep`	Makes the character with the tag look at the entity with the tag	`unitTag`: unit tag; `targetTag`: target tag; if several entities have the tag, the first one is chosen; `waitForCompletion`: wait for the turn to finish
`CheckAbilityUsedAtTargetStep` / `checkAbilityUsedAtTargetStep`	Checks whether the given ability was used on the given target (implemented only for attacks	`abilityIdent`: ability ident from the Abilities.json config. Works with part of the ability ident (i.e. if you enter Attack, the step triggers on any ability with Attack in its name); No description in code: `targetTag`

	and overwatch). Important: not all ability types are supported. If the step doesn't trigger, ask the developers to check whether the ability is covered by the step's logic.	
`CheckMetaCharacter` / `checkMetaCharacterStepData`	Checks that the given character is present, by name.	`name`: character name
`CheckObjectClickStep` / `checkObjectClickStep`	Waits for the required clickable object to be activated	`hoverName`: hover caption of the clickable object
`CheckObjectiveCompleteStep` / `checkObjectiveCompleteStep`	Checks whether the objective on the object with objectiveTag has been completed the number of times given in objectiveAmount	`objectiveTag`: tag of the object to check; `objectiveAmount`: number of times needed to consider the objective fully completed (can be 0)
`CheckPlayerChoicesStep` / `checkPlayerChoicesStep`	Checks that the given choices are present in or absent from the player's choices.	`playerHaveChoices`: list of choices the player must have made; `playerDoesntHaveChoices`: list of choices the player must NOT have made.
`CheckRescueUnitWithTagStep` / `checkRescueUnitWithTagStep`	Waits until the required number of units with the given	`unitsTag`: tag of the units to check; `percentToComplete`: share of rescued units required to complete the objective. 30% by default, set in the range from 0 to 1, where 0 is

	tag are "rescued" in one way or another	0% and 1 is 100%
`ClearAllCharacterMetaStatus` / `clearAllCharacterMetaStatusData`	Removes all statuses from characters, setting them to Ready	—
`CompleteMissionStep` / `completeMissionStep`	Ends the level with the player's victory	—
`CustomContent` / `customContentStepData`	Sets a custom set of characters in the character shop	`state`: content type: Default is regular, Rookies is newbies
`CutsceneZoneStep` / `cutsceneZoneStep`	Enables or disables a custom visibility zone for a cutscene.	`enable`: enable or disable.; `zone`: zone index
`DelayMetaStep` / `delayMetaStepData`	Adds a time delay in the hub before continuing.	`delayTime`: wait time in hours.; `delayKey`: timer identifier for the hub; `globalCounterKey`: if not empty, the wait value is taken from the global counter
`DelayStep` / `delayStep`	Adds a delay before the scenario continues.	`delay`: delay in seconds.
`DisableInteractiveCursor` / `disableInteractiveCursorStep`	If true, hovering over interactive objects, enemies, etc. is disabled. Otherwise it is enabled.	No description in code: `disable`
`EnableBuildingCutdown` / `enableBuildingCutdownStep`	Enables or disables hiding	`enable`: enable or disable.

	of buildings.	
`EnableLockCellStep` / `enableLockCellStepData`	Enables/disables blocking of a cell	`x,y,i`: cell coordinates; `blockTag`: blocker tag, so you don't have to list cells; `setLock`: true means the cell is blocked for the player; setLockAll: true means the cell is blocked for everyone, including the AI
`EnemyWithTagHasAttacked` / `enemyWithTagHasAttacked`	Whether the enemy with the given tag has attacked.	`targetTag`: enemy tag.
`ExplodeBombStep` / `explodeBombStep`	Detonates sabotage bombs by tag	`tag`: tag to detonate.
`FadeScreenStep` / `fadeScreenStep`	Fades the screen to black.	`fadeTime`: animation time in seconds.; `showFade`: shows the fade if true, hides it if false.
`FailMissionStep` / `failMissionStep`	End the mission with the player's defeat	—
`ForceAnimationPlayStep` / `forceAnimationPlayStep`	Starts playing an animation state on objects with the given tag. Doesn't wait for the animation to finish.	`animationStateName`: name of the animation state in the controller (check with the developers if needed); `targetTag`: tag of the objects that should play the animation; `startAnimationFromRandomPlace`: if true, the animation starts with a random time offset.; `useCrossFade`: if true, the new animation starts smoothly by blending the current and new animations.; `crossFadeDuration`: blend duration in seconds when useCrossFade is ticked; `dontSkip`: don't skip the animation with a skip (a fix for enemies on S02)
`ForceCompleteAITurn` / `forceCompleteAITurnStepData`	Makes AI-controlled factions skip their turn. Used to quickly return the turn to the player (at the end of a scenario before	`setEnabled`: if true, the AI skips turns until the ban is turned off by a step or the mission ends

	an autosave, for example). Important: if you use it at the end of a scenario, take control away from the player before this step (SwitchControls Step false). If you disabled it in the middle of a scenario, don't forget to enable it again.	
`ForceCompleteScenarioStep` / `forceCompleteScenarioStep`	Forcibly completes the given scenario.	`isSuccess`: marks the scenario as completed successfully.; `idToComplete`: ID of the scenario to complete.
`ForceSetLocationSoundAmbient` / `forceSetLocationSoundAmbientStepData`	Sets the ambient sound for the location	`daySoundTrigger`: daytime ambient sound; `nightSoundTrigger`: nighttime ambient sound
`ForceStopMetaTime` / `forceStopMetaTimeStepData`	Forcibly pauses time fast-forward in the hub	—
`ForceVisibleEntity` / `forceVisibleEntityStepData`	Enables/disables an object's visibility regardless of whether any of the Spartans can see it	`entityTag`: entity tag; `enable`: enable/disable forced visibility of the object
`GenericCounterStep` / `genericCountdownStepData`	A universal counter whose values are taken from the config.	`localizationKey`: localization key for display in the UI; `finalState`: the state the counter should have when it finishes counting; `counterKey`: unique text name of the counter; `showCounterInsteadOfTask`: show the counter icon instead of the objective in the UI; `turnsUpdateIcon`: name of the icon to show instead of the text; `showResultScreenText`: on the results screen, try to replace all {0} with

		human-readable values
`HelicopterAnimationStep` / `helicopterAnimationStepData`	Puts a character into the helicopter descent animation state.	`targetTag`: tag of the object to apply it to; `state`: the animation state to apply: PrepareLanding/Landing/Evacuation; `rotation`: angle to turn the character to; if -1, the character turns toward the map center; `delay`: delay before the action; No description in code: `waitForCompletion`
`HideAbilityStep` / `hideAbilityStepData`	Locks or unlocks an ability.	`abilityIdent`: ability name from the Abilities.json config.; `hide`: if true, the ability is locked; if false, unlocked
`HighlightAbilityButtonStep` / `highlightAbilityButtonStep`	Enables highlighting on a specific button in the ability panel. The highlight turns off only after the button is pressed. Exceptions: abilities that have the highlightable field in Abilities.json (for example, reload) use separate logic and are always highlighted when available.	`highlightIdent`: ability name from the Abilities.json config.
`HighlightCellStep` / `highlightCellStep`	Highlights the chosen cell for the player.	`X,Y`: coordinates of the cell to highlight; `floor`: the cell's floor, if the cell has several floors; `enableHighlight`: enable/disable the highlight.
`HubCustomUI` / `hubCustomUIStepData`	Controls the display of the HubWindow UI	`showTimePanel`: shows/hides the time fast-forward panel; `unlockNewsFeature`: once and permanently (through FeatureLockService) unlocks the news button, regardless of showNewsButton in other scenarios that may run later; No description in code: `showMapButton`,

		showHeadquartersButton, showBudgetButton, showCharactersShopButton, showContractsButton, showBarracksButton, showHospitalButton, showSaloonButton, showBuildButton, showNewsButton, showHoldButton, showMemorialButton, showHireCharacter, spawnMapMissions, showDaamatPower, showBlackMarket, showPolygon, ForceSetScreen
HubHideCityStep / hubHideCityStepData	Hides/shows cities for the player.	hideCities: hide or show cities
IsEntityInRadiusStep / isEntityInRadiusStepData	Triggers when a unit with the given tag is within the given radius of an object and/or character with the tag.	anchorTag: tag of the unit/object around which the check is done; targetTag: tag of the unit that must get within the radius; radius: radius
KillCharacterByTagStep / killCharacterByTagStep	Kills all characters with the given tag	No description in code: targetTag
LockAllAbilitiesButOneStep / lockAllAbilitiesButOneStep	Locks or unlocks all abilities except one.	unlockedIdent: ability name from the Abilities.json config. To lock all abilities, leave the string empty; unlockAbilities: if true, ALL abilities are unlocked
LockAllMissionAndTime / lockAllMissionAndTimeData	Locks time fast-forward and all missions on the map except one.	ExceptIdMission: which mission stays available for selection.; Enable: enable or disable the mode.
LockCallEvacuationStep / lockCallEvacuationStepData	Locks/unlocks the ability to call the evacuation zone.	No description in code: setLocked
LockCharacterSelectionStep / lockCharacterSelection	Selects the character given by tag and	unlockSelection: true unlocks character selection; targetTag: tag of the character to

`Step`	forbids selecting others. All of the player's characters have the tags PlayerCharacterN, where N is a sequence number from 1 to 4. When unlocking, specify the character to select afterwards, otherwise the selection may be reset or a mercenary without action points may be selected. Important: when unlocking character selection, you must specify the index of the character to select after unlocking. If you don't, the selection may be cleared from all characters or a character who has already spent all action points may be selected.	select
`LockDoorStep` / `lockDoorStep`	Locks/unlocks all doors with a certain tag	`doorTag`: door tag; `setLocked`: lock/unlock the door.; `isInstant`: open the door instantly

`LockEntityRotate` / `lockEntityRotateData`	Locks or unlocks a character turning toward enemies	No description in code: `isLock`, `targetTag`
`LockMapGenericModuleStep` / `lockMapGenericModuleStepData`	Locks/unlocks all mining/demining objects	`tag`: tag; `setActive`: lock/unlock
`LockTurnTransitionStep` / `lockTurnTransitionStep`	Locks/unlocks passing the turn to another faction.	`setLocked`: whether to lock passing the turn
`MakeCharacterUseAbilityAtTargetStep` / `makeCharacterUseAbilityAtTargetStep`	Makes all characters with the tag use the given ability on the target with the tag. Important: at this stage the step supports only abilities applied to a specific unit.	`targetTag`: target tag; `unitTag`: tag of the unit that should use the ability; `abilityIdent`: ability ID from the Abilities.json config
`MoveCharacterWithTagToPointStep` / `moveCharacterWithTagToPointStep`	Gives a character a point to move to. Important: while the player's character is moving, the player has no cursor control but can control the camera and use abilities. Better not to use it in gameplay, only for staging scenes.	`isTeleport`: teleport the target instead of running; `targetTag`: target tag; `positionX`, `positionY`: point coordinates (can be seen in the cell name in the scene); `floor`: the cell's floor, if the cell has several floors (select the cell in the scene to see it; the cell name has the format [X:Y]:floor; `waitForCompletion`: whether to wait for the movement to finish before completing the step

`MoveObjectsWithTagToUnitCellStep` / `moveObjectsWithTagToUnitCellStep`	Moves objects with targetTag to the cell of the unit with unitTag. The order is not guaranteed: the step takes the first random object and moves it to the first random suitable unit by tag.	`unitTag`: tag of the units to move the object to; `targetTag`: tag of the objects to be moved to the units
`MoveObjectTransformWithTagToPointStep` / `moveObjectTransformWithTagToPointStep`	Moves an object to the given point in a straight line without pathfinding. Movement works like the standard character movement, with smooth acceleration at the start and smooth deceleration at the end	`isTeleport`: teleport the target instead of smooth movement; `targetTag`: target tag; `X, Y, Z`: world coordinates of the point (can be seen in the cell's Cell Center in the scene); `waitForCompletion`: whether to wait for the movement to finish before completing the step; `maxSpeedOverride`: if set to a number greater than zero, it is used as the character's maximum speed; otherwise the character's default maximum speed is used
`NoAliveEntitiesWithAbilityStep` / `noAliveEntitiesWithAbilityStep`	Checks whether living objects with abilities of the given types remain on the map. It can also check whether the target with vipTag has a buff of the chosen type. If it does, the step is NOT	`targetTag`: target tag; `abilities`: list of ability types; `requiredStatusEffectsSubstrings`: list of substrings that may occur in the names of status effects applied by the character's abilities; `vipTag`: tag of the VIP target. Optional field, can be left empty.; `vipBuffType`: the buff whose presence is checked on the VIP. If the buff is present, the step is NOT completed

	completed.	
`NoAliveEntitiesWithTagStep` / `noAliveEntitiesWithTagStep`	Checks whether living objects with the given tag remain on the map.	`targetTag`: target tag; `ignoreStunned`: don't count stunned or restrained objects; `onlyActualDeath`: consider objects alive regardless of state while IsAlive is true
`OpenDoor` / `openDoorStepData`	Opens or closes all doors with the given tag.	`doorTag`: door tag; `isOpen`: open or close the door; `isInstant`: open the door instantly
`PlantBombStep` / `plantBombStepData`	Plants a bomb in the cell next to the object.	`targetTag`: object tag; `explosionRadius`: explosion radius; `explosionDelay`: delay before the explosion; `explosionPrefab`: explosion visual from the "paths.json" config; `showExplosionTimer`: show a plate with "Until explosion: X"; `explosionTime`: turns until the explosion; `timerIconID`: icon from the "paths.json" config; `timerTextID`: timer text; `timerYOffset`: timer position offset relative to the ground; `showExplosionRadius`: highlight the cells the explosion will hit; `explosionRadiusView`: name of the prefab from the "paths.json" config used to highlight the cells; `dontUseTransition`: don't move the camera; `dontExplodeAfterTimer`: don't explode when the timer runs out
`PlayDialogueStep` / `playDialogueStep`	Starts the chosen dialogue. If waiting is enabled, the scenario won't continue until the dialogue finishes completely. Important: if the dialogue config has no audio file id, don't tick the box!:	`id`: dialogue ID in the dialogue config (Assets/StreamingAssets/Configs/Dialogues.json); `waitForCompletion`: whether to wait for the dialogue to end (yes by default).; `showSubtitles`: show the dialogue subtitles.

`PlayerChoiceStep` / `playerChoiceStep`	Shows the player a story choice window and waits for the player to choose	`choiceId`: id of the story choice from PlayerChoice.json
`PlaySoundStep` / `playSoundStep`	Plays a sound by ID. Doesn't wait for it to finish (the scenario continues right after the sound starts)	`id`: sound id in the sound config; No description in code: `objectIdent`
`PlaySpawnScenarioStep` / `playSpawnScenarioStep`	Starts a scenario that makes units appear in the scene. Important: the scamp won't turn back on by itself. To turn it back on, use ScampSwitchStep.	`scenarioId`: id of the spawn scenario in SpawnScenarioTemplate in the scene; `spawnImmediately`: if true, the scenario starts and the first wave spawns immediately; if not, the first wave spawns at the end of the player's turn; `lockScampBeforeSpawn`: if true, turns off the scamp logic right before units start spawning; No description in code: `doNotFocusCamera`
`PlayStoryMusicStep` / `playStoryMusicStepData`	Plays music by the musicEvent event name. Stored in the storyMusic field in MusicController	No description in code: `musicPair`, `musicEvent`
`PlayVideoCall` / `playVideoCallStepData`	Starts playing a video call; video calls are added through Choice	—
`PointCameraStep` / `pointCameraStep`	Points the camera at an object by tag.	`targetTag`: tag; `height`: target camera height; `angle`: target camera rotation angle (from 0 to 360 degrees); `transitionTime`: time for the camera to reach the given position;

		`waitForCompletion`: whether to wait for the transition to finish before continuing the scenario (if true, the scenario continues only after the transition ends and the camera stops completely); `enableObjectXray`: whether to keep object X-ray on during the camera transition. If false, all transparent objects become visible at the start of the transition.; `enableFloorCutdown`: whether to keep floor hiding on during the camera transition. If false, all hidden floors are shown at the start of the transition.
`ReloadStep` / `reloadStep`	Reloads weapons for all Spartans	—
`RemoveAllStatusEffectsByTagStep` / `removeAllStatusEffectsByTagStep`	Removes all status effects from characters with the given tag.	`targetTag`: target tag
`RemoveUIStep` / `removeUIStep`	Removes a step from the objective list in the UI by localizationKey. Important: when the key of an existing ShowUIStep changes, all RemoveUIStep steps linked to it are NOT UPDATED AUTOMATICALLY; go through all such steps with the old localization path and update them by hand	`isCounter`: if true, the step tries to remove the step counter that appeared after WaitForNumberOfSteps started (to the right of the objectives panel); No description in code: `localizationKey`
`RequestBarrackDialogue`	Requests	`Id`: dialogue Id in the

/ `requestBarracksDialogueData`	barracks dialogues by Id for later playback	Assets/StreamingAssets/Configs/BarracksDialogue/BarracksDialogueConfig.json config
`RescueAdditionalUnitWithTagStep` / `rescueAdditionalUnitWithTagStep`	Waits until the number of units with the given tag is 0. Also adds and updates the objective in the objective list and manages the rescued counter	`unitsTag`: tag of the units to check; `uiTaskString`: localization key for the objective in the UI. IMPORTANT: the key must have the format Some text {0}/{1}. This is needed for the counter to display correctly in the UI; `mainUi`: as with ShowUIStep, shows the objective among the main objectives (if true) or the optional objectives of the combat UI; No description in code: `uiTaskKey`
`RescueUnitWithTagStep` / `rescueUnitWithTagStep`	Waits until the required number of units with the given tag are "rescued" in one way or another. Also adds and updates the objective in the objective list and manages the rescued counter	`unitsTag`: tag of the units to check; `percentToComplete`: share of rescued units required to complete the objective. 30% by default, set in the range from 0 to 1, where 0 is 0% and 1 is 100%; `uiTaskString`: localization key for the objective in the UI. IMPORTANT: the key must have the format Some text {0}/{1}. This is needed for the counter to display correctly in the UI; `mainUi`: as with ShowUIStep, shows the objective among the main objectives (if true) or the optional objectives of the combat UI; No description in code: `uiTaskKey`
`RestrictOverCell` / `restrictOverwatchCellStep`	Enables/disables overwatch triggering on certain cells	`x,y,i`: cell coordinates; `setEnable`: true means overwatch doesn't trigger on this cell
`ReturnCameraToSelectedCharacterStep` / `returnCameraToSelectedCharacterStep`	Returns the camera to the selected character (selects it again)	—
`ReviveCharactersByTagStep` / `reviveCharactersByTagS`	Revives all characters with	`targetTag`: target tag; `hp`: health

tep	the given tag and sets their health to the chosen amount (but no less than 1).	
RotateObjectsWithTagOnAngleStep / rotateObjectsWithTagOnAngleStep	Rotates objects with the tag to the given Euler angle around Y	targetTag: tag of the objects to rotate; angle: Euler angle around Y to rotate the object to
ScampSwitchStep / scampSwitchStep	Turns the scamp logic off or on. After turning on, it creates scamp events, so if the player's units are in the enemies' field of view (and it is the player's turn when it is turned on), the scamp fires. Important: if you turned it off, don't forget to turn it back on	No description in code: setEnabled
SetAIPriorityTargetStep / setAIPriorityTargetStepData	Sets a priority target for the AI	priorityTargetTag: tag of the target the AI will head for; workerAmount: number of AI units that can be given the task; isCompletable: whether the task can be completed, or the AI will work on it indefinitely (for example, defending an object); taskRadius: task radius. For all interaction tasks (killing a VIP, (de)mining, etc.) it is 0; forceToComplete: forcibly completes a task that can't be completed normally (isCompletable == false)
SetCharacterActionPointStep / setCharacterActionPointStep	Sets the action, attack and move points of characters with	targetTag: target tag; commonPoints: action points; attackPoints: attack points; movePoint: move points; No description in code: movePoints

	the given tag.	
`SetCharacterRandomStatus` / `setCharacterRandomStatusData`	Assigns random states to all ready mercenaries after combat.	—
`SetCharacterToPlayerByTagStep` / `setCharacterToPlayerByTagStep`	Adds a character to the player's squad or removes it	`targetTag`: target tag; `setToPlayer`: if true, adds the character to the squad; if false, removes it from the squad.
`SetFactionProfileStep` / `setFactionProfileStepData`	Sets a faction profile.	`profileId`: profile ID from the FactionProfilesConfig.json config.
`SetNewsProfile` / `setNewsProfileStepData`	Sets the random news profile (stage).	`profileId`: profile ID
`SetPlayerTargetsByTagStep` / `setPlayerTargetsByTagStep`	Lets you restrict the targets of the player's characters to the set of objects with the given tag	`targetTag`: target tag; `restrictTarget`: allows aiming ONLY at this target; `removeLimitation`: if true, removes the restrictions
`SetTagStep` / `setTagStepData`	Adds a tag to an object or removes it.	`targetTag`: tag of the target to which the tag is added or from which it is removed; `isTagAdded`: if true, adds the given tag; if not, removes it; `tagToAddOrRemove`: the tag to add or remove
`SetUnitFactionStep` / `setUnitFactionStepData`	Changes the faction of the character with the tag.	`targetTag`: character tag; `faction`: new faction
`ShowCinematicStep` / `showCinematicData`	Starts the cinematic with the chosen Id. X,Y,Z are the coordinates of the cinematic prefab	`targetTag`: tag of the object the cinematic is positioned by; `showSubtitles`: show subtitles; `continueCinematicSequence`: if true, this cinematic is followed by a cutscene or cinematic and a seamless transition between them is needed (the cutscene UI won't be hidden and gameplay elements won't be enabled). If unticked, the cinematic ends as

		usual; No description in code: `cutsceneId`, `X`, `Y`, `Z`, `rotX`, `rotY`, `rotZ`
`ShowCreditsFinal` / `showCreditsFinalData`	Starts the final credits	—
`ShowHintMarkStep` / `showHintMarkStep`	Shows or hides a hint marker with an icon.	`targetTag`: tag of the target the marker attaches to. The target must be static.; `spritePath`: path to the icon sprite from Paths.json. When adding a new sprite, add it to Addressables, add its path to Paths.json in the sprites section with the name as the key, and put the KEY in this field.; `yOffset`: height offset of the marker relative to the target position, in meters.; `show`: show/hide the marker. To hide the marker, the target tag must be given; the other fields can be left empty
`ShowMissionMarkerStep` / `showMissionMarkStep`	Shows or hides a mission objective marker with an icon.	`targetTag`: tag of the target the marker attaches to.; `spritePath`: path to the icon sprite from Paths.json. When adding a new sprite, add it to Addressables, add its path to Paths.json in the sprites section with the name as the key, and put the KEY in this field.; `yOffset`: height offset of the marker relative to the target position, in meters.; `show`: show/hide the marker. To hide the marker, the target tag must be given; the other fields can be left empty
`ShowModalWindowStep` / `showModalWindowStep`	Shows a modal window by id from the ModalWindows Configs.json config	No description in code: `id`
`ShowUIStep` / `showUIStep`	Displays an objective in the UI objective list. Important: when an existing ShowUIStep changes, all UpdateUIStep and	`localizationKey`: localization key for the objective text; `main`: list the objective among the mission's main objectives or the optional ones; No description in code: `isCounter`, `isDaysTimer`, `currentValue`, `maxVale`, `timerValue`

	RemoveUIStep steps linked to it are NOT UPDATED AUTOMATICALLY; go through all such steps with the old localization path and update them by hand	
`ShowVideo` / `showVideoStepData`	Starts a video cutscene	`VideoId`: video file id from Paths.json; `SoundId`: sound event id
`SpartanPickUpTargetStep` / `spartanPickUpTargetStep`	Waits until a Spartan picks up the target with the given tag	No description in code: `targetTag`
`SpawnMissionOnMap` / `spawnMissionOnMapStepData`	Creates a mission of the given type on the global map.	`zoneId`: zone to create the mission in; `missionType`: type of mission to create, from the MissionType enum; `category`: mission category from the MissionCategoty enum; `id`: mapId (for active missions) or id (for story passive missions); `spawnPointId`: id of the point to create the mission at (leave empty to choose randomly)
`StartCutsceneCameraStep` / `startCutsceneCameraStep`	Switches the camera to the CutsceneCamera with the given tag	`cameraTag`: tag of the camera prefab in the scene. Must be unique within the scene; `continueCinematicSequence`: if true, this cutscene is followed by another cutscene or cinematic and a seamless transition between them is needed (the cutscene UI won't be hidden and gameplay elements won't be enabled). If unticked, the cutscene ends as usual; `waitForCompletion`: whether the step waits for the camera movement to finish before moving to the next step; `parentTag`: assigns a parent to the camera target. If you don't need to follow the target, better leave it empty.; `dontSkip`: don't skip the animation with a skip

`StartLocationCheckStep` / `startLocationCheckStep`	Waits for units with the given tag to be in the area with the given tag and stay there for the given number of turns. Also adds and updates the objective in the objective list and manages the remaining-turns counter in the combat UI	`id`: step ID. Must be unique within the mission. IMPORTANT: when changing the ID in a StartLocationCheckStep, you must update the ID in the linked WaitForLocationCheckStep. To do this: change the ID in StartLocationCheckStep, save the scenario, go to the WaitForLocationCheckStep, update the ID there and save the scenario again.; `removeAfterComplete`: if set, the service removes the objective from the objective list itself when it finishes; `unitsTag`: tag of the units to check; `triggerTag`: trigger tag. A collider must be set up on it and the scene saved. The trigger's cells are used for the check. They can be seen in MapEntityBehaviour in the BaseCellForScenarios field; `targetUnitsNumber`: number of units that must be in the area for the turn countdown to start. If 0, the service waits for ALL LIVING units with the tag to be in the area; `countKnockdown`: whether to count critically wounded units; `turnsToWait`: number of turns to wait in the area after all required units have gathered there; `uiTaskString`: localization key for the objective in the UI. IMPORTANT: the key must have the format Some text {0}/{1}. This is needed for the counter to display correctly in the UI; `mainUi`: as with ShowUIStep, shows the objective among the main objectives (if true) or the optional objectives of the combat UI; `turnsUpdateKey`: localization key for updating the dedicated UI that shows the turns left until the event.; `turnsUpdateIcon`: icon key from Paths.json to show in the remaining-turns counter.; `turnsUpdateZeroTurnsLeftLocalizationKey`: localization key for the text shown when the turn counter runs out, before it is hidden; No description in code: `uiTaskKey`, `uiTaskMissionResultsKey`
`StartMissionsSpawnPipelineStep` / `startMissionsSpawnPipelineStepData`	Starts issuing missions	`startActives`: starts issuing active missions; `startPassives`: starts issuing passive missions
`StopSoundStep` /	Stops the	`id`: sound identifier; `customPrefix`: used when

`stopSoundStep`	chosen sound	the event sent is something other than Stop
`StrategicOffensive` / `strategicOffensiveStepData`	Enables the strategic offensive and waits for it to end; mission count settings are in DaamatPowerConfig.json	—
`SwitchControlsStep` / `switchControlsStep`	Enables/disables gameplay controls.	No description in code: `enabled`
`SwitchFogStep` / `switchFogStep`	Toggles the fog of war state on the map.	No description in code: `setEnabled`
`SwitchObjectsByTagStep` / `switchObjectsByTagStep`	Enables/disables the GameObject with the given tag. Important: the object must initially be ENABLED, otherwise it won't initialize and the step won't work on it. I.e. if the object must be disabled from the start and enabled later, first disable it with a step and then enable it with a step. You MUST NOT disable it by hand in the scene initially	`targetTag`: target tag; `setEnabled`: the desired object state.
`SwitchRandomReplicasStep` /	Enables/disable	No description in code: `enable`

`switchRandomReplicasStep`	s playback of random character lines.	
`SwitchTiedDownStepData` / `switchTiedDownStepData`	Toggles the tied-up animation for VIP characters being rescued.	`targetTag`: target tag; `isTiedDown`: if true, the animation with the character sitting tied up is enabled; otherwise, the release animation
`SwitchWeaponVisibilityStep` / `switchWeaponVisibilityStepData`	Toggles the visibility of the weapon in the given character's hands.	`targetTag`: target tag; `isWeaponVisible`: if true, the weapon is visible; otherwise invisible
`ToggleInteractiveObjectsStep` / `toggleInteractiveObjects`	Enables or disables the given interactive objects.	`targetTag`: interactive object tag; `isEnabled`: if ticked, enables the interactive object; otherwise disables it
`TriggerInRangeStep` / `triggerInRangeStep`	Triggers when the trigger with the given tag ends up on one of the cells occupied by the target's collider. The cells are updated when the map is saved and are stored in the BaseCellsForScenarios field. IMPORTANT: the object's pivot must be ABOVE the cell it occupies (don't sink colliders underground).	`targetTag`: target tag
`TutorialMarker` /	Shows a	`rectTransformName`: exact name of the object

`tutorialMarkerStepData`	tutorial marker on a button	above which the marker is shown; `isShow`: show or hide the marker; No description in code: `rotation`, `type`
`UnitInSight` / `unitInSight`	Triggers when an enemy unit with the given tag comes into the field of view of one of the Spartans.	`targetTag`: enemy unit tag
`UnitWithTagHpCheckStep` / `unitWithTagHpCheckStep`	Checks the health of the character with the given tag and completes when the condition is met.	`targetTag`: target tag; `checkAllTargets`: if true, the step completes only when ALL entities with the given tag meet the condition. If unticked, the step completes if ANY entity meets the condition.; `relation`: condition: "==", "!=", "<=", ">=", ">", "<"; `targetHp`: health value to compare with
`UnlockBlockerByTagStep` / `unlockBlockerByTagStep`	Removes blockers with the given tag from the map.	No description in code: `targetTag`
`UpdateCounterStep` / `updateCounterStepData`	Changes the value of a named counter.	`isSet`: if true, sets the value; if false, adds it; `key`: counter key; `value`: the counter value itself
`UpdateCoverCellState` / `updateCoverStep`	Enables/disables cover on a cell. After enabling, updates characters so they take it	`x,y,i`: cell coordinates; `setEnable`: true means the cover on the cell works, false means it is disabled
`UpdateUIStatusStep` / `updateUiStatusStep`	Updates an objective's status in the UI objective list. Important: when an existing ShowUIStep changes, all UpdateUIStep	`localizationKey`: localization key for the objective text; `isSuccess`: mark the objective completed or failed; `clipParameters`: cuts all substituted parameters out of the text. Needed when the step is used to update an objective created through StartLocationCheckStep; `UpdateInsteadOfAdd`: tries to update the line instead of adding it if it is missing; `isMissionTimer`: the step belongs to the mission timer (an objective created through

	steps linked to it are NOT UPDATED AUTOMATICALLY; go through all such steps with the old localization path and update them by hand	GenericCounterStep). If mission timers are turned off in the settings, the step is skipped so as not to show an objective that was never shown; No description in code: `state`, `skipAnimations`, `isCounter`, `counter1`, `counter2`
`WaitAnimationEventStep` / `waitAnimationEventStepData`	Waits for an animation event from a cutscene.	`cinematicTag`: cutscene tag.; `animationEventIndex`: event index (in order on the animation track
`WaitAnyCharacterWithStatus` / `waitAnyCharacterWithStatusStep`	Waits for a character with a certain status (from CharacterMetaStatusType) to appear at the base, then completes.	`waitStatus`: the expected character status. Use Injured if you want to track any wound
`WaitBuildStep` / `waitBuildStepData`	Waits for a compartment's construction to start/finish	`buildId`: ID of the compartment to wait for; `isComplete`: if true, waits for the compartment's construction to finish
`WaitCharacterBuy` / `waitCharacterBuyStepData`	Waits for a character to be bought, then completes	—
`WaitCompleteBarrackDialogue` / `waitBarracksDialogueCompletedStepData`	Waits for the barracks dialogues to end	`Count`: number of dialogues that must be listened to; `DialogId`: identifier of the dialogue being listened to
`WaitContract` / `waitContractStepData`	Waits for a contract	`contractId`: identifier of the contract to wait for; `isComplete`: if true, waits for this contract to be completed
`WaitCounter` / `waitCounterCompleteSte`	Lets you wait for a certain	`counterKey1`: value of counter 1; `counterKey2`:

`pData`	counter to finish	value of counter 2
`WaitDeminingAbilityCompleteStep` / `waitDeminingAbilityCompleteStepData`	Waits until the demining ability has worked for the required number of turns.	—
`WaitForCharacterWithTagEvacuationStep` / `waitForCharacterWithTagEvacuationStepData`	Waits until the character with the given tag is evacuated	`targetTag`: target tag
`WaitForCutsceneCompleteStep` / `waitForCutsceneCompleteStep`	Waits for a cutscene/cinematic to end.	—
`WaitForFactionTurnStep` / `waitForFactionTurnStep`	Waits for the given side's turn to start, or completes at once if it is that side's turn now.	`factionIdx`: 1 is the player, 2 AI-controlled allies, 3 enemies, 4 civilians
`WaitForFirstKillStep` / `waitForFirstKillStepData`	Waits for the first character of the given faction to die	`faction`: the faction to wait for.
`WaitForLocationCheckStep` / `waitForLocationCheckStep`	Waits for the task checking that units are in the given area to complete.	`taskId`: task id
`WaitHideLoadScreen` / `waitHideLoadScreenData`	Waits for the LoadScreen to hide, then closes.	—
`WaitLootSpawn` / `waitLootSpawnData`	Waits for a loot bag to appear	—
`WaitMetaDateTime` / `waitMetaDateTimeData`	Waits for a certain date on the in-game	`day`: day; `month`: month; `year`: year

	clock	
`WaitMissionStartStep` / `waitMissionStartStepData`	Waits for a mission to start/end	`missionType`: type of the mission to wait for: Passive or Active; `missionTypes`: when many different mission types need to be checked, separated by ;; `id`: if not empty, waits for an active mission or a passive story mission with a certain MapId; `isComplete`: if true, waits for the mission to end; false if waiting for it to start
`WaitNumberOfTurnsStep` / `waitNumberOfTurnsStep`	Waits for the given number of turns	`turns`: how many turns to wait; `showCounter`: if true, the "Events" UI window shows a counter of the turns left until the wait ends; `localizationKey`: localization key for the turn counter tooltip; `turnsUpdateIcon`: name of the icon from the TMP_Sprite Asset TurnTimerTexturePacker1 to show in the remaining-turns counter; `turnsUpdateZeroTurnsLeftLocalizationKey`: localization key for the text shown when the turn counter runs out, before it is hidden
`WaitScreenStep` / `waitScreenStepData`	Waits until the given base screen is opened.	`screenID`: screen ID (see InteractiveEnvironmentType.cs)
`WaitSpawnStorm` / `waitSpawnStormStepData`	Waits for a storm to be created on the map	—
`WaitStoredEquipment` / `waitStoredEquipmentStepData`	Waits for the given item to appear in the base storage.	`itemId`: item identifier
`WaitWindowClose` / `waitWindowClosedData`	Waits for the required window to close	No description in code: `WindowId`