IamInnocent's ZH Feature Implementation

List Updated: 23/01/26

CustomTypes:

- Custom Damage Types:

```
{ "CustomDamageTypes",                    INI::parseCustomDamageTypesDefinition },
```

```
{ "CustomDamageType", parseCustomDamageType, NULL, 0 }
{ "Damage", parseDefaultDamage, NULL, 0 }
{ "LinkDamageType", parseLinkDamageType,    NULL, 0 },
{ "LinkCustomDamageType", parseLinkCustomDamageTypes, NULL, 0 }
```

- Added CustomDamageTypes for WeaponSets and ParticleCannonLasers.
- The CustomDamageTypes are separated from DamageTypes. CustomDamageTypes can be configured to have their own Default (%) value, LinkedDamageType, and LinkedCustomDamageTypes.
- LinkedCustomDamageTypes for CustomDamageTypes searches for any other CustomDamageTypes first (in order) if the armor does not have any declared verses for the current CustomDamageTypes.
- If none of the CustomDamageTypes were found when calculating the Damage Coefficient, the system searches for the LinkedDamageType for calculating damage. LinkedDamageType considers one of the DamageTypes for calculating damage.
- If LinkedDamageType is set to 'Default' it considers the 'Default' Armor's Damage Coefficient for damage calculation, or a custom Damage Cofficient (%) (See below).
- You could set a Damage for determining a constant Damage Coefficient for damage calculation. Think of it as a Default damage type, but universal for one CustomDamageType. (This is very useful if you want to set 0% for universal, but 99999% for one unit.)
- If a CustomDamageType is set with none of the above parameters configured, then it will use its own DamageType for the damage calculation.

For Example, declaring in Armor.ini:

```
CustomDamageTypes
        CustomDamageType = SNIPERXL
        Damage = 333%
        LinkDamageType = SNIPER
        CustomDamageType = SNIPERXXL
        Damage = 3%
        LinkDamageType = SNIPER
        LinkCustomDamageTypes = SNIPERXL SNIPERL
        CustomDamageType = CustomSNIPER
        LinkCustomDamageTypes = SNIPERXXL
End
```

SNIPERXXL and CustomDefault will both register as the global custom damage type. However, units can also register their separate custom damage types for damage calculation.

- How it works: When a weapon has CustomSNIPER as its CustomDamageType and the Object registers Armor of SNIPERXL at 65% but does not register CustomSNIPER, the damage will consider SNIPERXL for damage calculation. This is because CustomDefault has SNIPERXXL within its LinkCustomDamageTypes and SNIPERXL is within SNIPERXXL's LinkCustomDamageTypes.
- However, when a weapon has SNIPERL as its CustomDamageType, the weapon will not consider its CustomDamageType for damage calculation because it's not declared in both the global CustomDamageTypes and the unit's CustomArmor.

> When registering the Armor for a unit, parseArmorCoefficients has been modified so when you try to register the DamageType Name, it will first consider the existing Damage Flags. If it doesn't exist in the Damage Flags, it will register as a CustomArmor for the unit towards the CustomDamageType.

- You can also register the CustomArmor without declaring at the GlobalData and use it exclusively for a Unit.

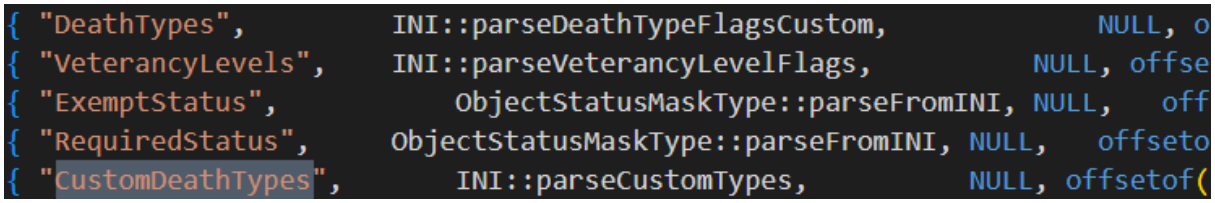
Armor BattleBusTruckArmor

Armor = SuperBattleBusExclusiveDestroyer 10000%

End

- This means a weapon with CustomDamageType of SuperBattleBusExclusiveDestroyer will trigger the declared damage values configuration, even without declaring SuperBattleBusExclusiveDestroyer at CustomDamageTypes.

- Custom Death Types:



- Added CustomDeathType for WeaponSets and ParticleCannonLasers.
- CustomDeathTypes are separated from DeathTypes. If a ModuleTag does not have CustomDeathType configured, then they will be using their DeathType.
- To assign Custom Death Types for a Module, use CustomDeathTypes. “ALL” or “NONE” are carried from DeathTypes, and they cannot be reassigned to CustomDeathTypes. +CUSTOMDEATH1 -CUSTOMDEATH2 will work the same way as DeathTypes to CustomDeathTypes.

- Custom Status Types:

- Implemented Custom Status Types which function as a form of Dynamic Status Types.
- Custom Status Types *do not* need to be declared; they are automatically registered after parsing from the functions below.
- Custom Status Types *do not* affect the Object’s actions in any way and are independent from existing StatusTypes.
- Custom Status Types currently can be used to affect the Object’s Tint Status and Firing Tracker Bonus conditions. (See below & the changes for Weapons.)
- Feature inclusive Custom Status:
  - “STOP_ATTACKING” - Prevents ongoing Objects from Attacking. (Does not stop the Object from trying to acquire a target. For that matter, use “NO_ATTACK” StatusType instead.)

- “AIM_NO_ATTACK” - Prevents Objects from Firing their Weapons but still allows Objects to acquire targets.
- “ZERO_DAMAGE” - Prevents Objects from Acquiring Targets or Firing Weapons and makes their Weapons deal 0 damage. Sets the cursor to ‘Generic Invalid’ or ‘SCCNoAction’ when trying to attack. (Same feature as Weapons with 0 damage or the traditional ‘Hold Fire’ Button seen in mods.)
- “NO_DAMAGE” - Allow Objects to Acquire Targets and Fire Weapons, but deals no damage.

Name	Status Type		Stops User from Targeting Objects with Weapons	Stops User from Acquiring Targets (Passively)	Tells User to Stop Firing Weapons on Targeting	Prevents User from Firing Weapons	Stops User from showing Attack Cursor	Shows ‘Stop’ Cursor when Targeting Objects	Shows ‘Stop’ Cursor when Targeting Ground	Stops User from Dealing Weapon Damage
	Original	Custom								
NO_ATTACK	✓		✓	✓		✓	✓			
No Weapons	-	-	✓	✓	✓	✓	✓	✓	✓	✓
0 Damage Weapon	-	-	✓							✓
-9999% Weapon Bonus Damage	-	-	✓							✓
STOP_ATTACKING		✓	✓		✓					
AIM_NO_ATTACK		✓				✓				
ZERO_DAMAGE		✓	✓	✓	✓	✓	✓	✓	✓	✓
NO_DAMAGE		✓								✓

- “SHIELDED_TARGET” - Redirect Weapons fired towards affected units onto the Status Giver. This triggers while weapons are fired onto the affected units. Works like Black Hole Armor from RA3.
- “DISABLED_MOVEMENT” - As implied, units affected by this status cannot move, but can still do other actions such as attacking. This is similar to Stunned status that an Object has when it is very high above the altitude.

- CustomWeaponBonuses:

```
{ "WeaponBonus",                          WeaponBonusSet::parseWeaponBonusSetPtr,
{ "CustomWeaponBonus",                WeaponBonusSet::parseCustomWeaponBonusSetPtr,
```

- Global Variable, declared in GameData.ini. For variables exclusive to a Weapon, check the Weapons Section.
- Operates like WeaponBonus but can be either declared globally or within a weapon.
- It can only declare for CustomStatusTypes.
- Declared by using the same format as WeaponBonus.
- Can be declared separately alongside WeaponBonus.

- Custom Radius Cursors:

```
{ "CustomRadiusCursor",         INI::parseCustomRadiusDecalDefinition },
```

- Added CustomRadiusCursor, declared outside InGameUI.
- Declared in the same format as RadiusCursors but with its Unique Template Name.
- To use them, declare the Template Name under the CustomRadiusCursorType in the CommandButton (See CommandButton Section below.)
  NOTE: Do not declare them within InGameUI, declare it outside the InGameUI.
- Declared in the same format as RadiusCursors but with its Unique Template Name.

For example,

```
  AmbulanceRadiusCursor
    Texture         = SCCAmbulance
    Style           = SHADOW_ALPHA_DECAL
    OpacityMin      = 50%
    OpacityMax      = 100%
    OpacityThrobTime  = 1000
    Color           = R:240 G:240 B:240 A:255
    OnlyVisibleToOwningPlayer = Yes
  End

End

CustomRadiusCursor TestSpectreCustomRadius
    Texture         = SCCSpecterTest
    Style           = SHADOW_ALPHA_DECAL
    OpacityMin        = 55%
    OpacityMax        = 75%
    OpacityThrobTime  = 500
    Color             = R:222 G:177 B:122 A:255
    OnlyVisibleToOwningPlayer = Yes
End
```

The CustomRadiusCursor is declared outside the InGameUI and its Template Name be used under CustomRadiusCursorType of CommandButton.

- TintCustomStatus:

```
{ "UseVanillaDiagonalMoveSpeed",          INI::parseBool,       NULL,           offsetof(Global
{ "TintStatus",  GlobalData::parseTintStatusType, NULL, offsetof(GlobalData, m_colorTintTypes)
{ "TintCustomStatus",    GlobalData::parseTintCustomStatusType, NULL, offsetof(GlobalData, m_c
```

- Global Variable, declared in GameData.ini. Sets the Tint or Color for Custom Status Types.
- Functions as a TintStatus for CustomStatusTypes.
- Declared by using the same format as TintStatus.
- Can be declared separately alongside TintStatus.

General:

Declared in GameData.ini:

- Firing Tracker Weapon Bonuses:

```
{ "FiringTrackerWeaponBonusStatus",              GlobalData::parseTrackerWeaponBonusStatus,  NULL
{ "FiringTrackerCustomWeaponBonusStatus",        GlobalData::parseTrackerCustomWeaponBonusStatus,
```

- Added Mechanics that work like Avenger's Designator. But more robust as the Bonus is declared before attack state and cleared after attack state.
- Global Variable, declared in GameData.ini. For variables exclusive to a Weapon, check the Weapons Section.
- It works by attacking a target that is affected by a StatusType. If the target has a StatusType or one of the StatusTypes declared, it grants the attacking Unit a WeaponBonus condition, but remove it when targeting a Location or if the unit doesn't have the affected StatusType.
- Register the First Variable as the WeaponBonusConditionType, then following the StatusTypes and CustomStatusTypes to indicate the Statuses required for granting the bonuses.
- This feature is compatible with WeaponBonusUpdate. WeaponBonusUpdate registers the bonus granted to a separated variable so that the same Bonuses don't get removed by Firing Tracker Weapon Bonuses.

Example,

FiringTrackerWeaponBonusStatus = FRENZY_ONE CUSTOMSTATUS1

FiringTrackerCustomWeaponBonusStatus = CUSTOMBONUS1 TARGET_FAERIE_FIRE CUSTOMSTATUS1 IGNORING_STEALTH CUSTOMSTATUS2

- This means that when attacking a Unit with IGNORING_STEALTH, it will grant FRENZY_ONE to the attacking unit. While attacking units affected by TARGET_FAERIE_FIRE, CUSTOMSTATUS1, IGNORING_STEALTH, or CUSTOMSTATUS2, it will grant CUSTOMBONUS1 to the unit.
- The bonuses are also removed when targeting a Location or if the target does not have the affected StatusTypes, unless the attacker is affected by WeaponBonusUpdate that grants the same WeaponBonuses.

- Countermeasures Behavior:

```
{ "CountermeasuresDetonateNonTrackingMissiles",  INI::parseBool, NULL, offsetof(GlobalData,
```

- Added CountermeasuresDetonateNonTrackingMissiles. If set to ‘Yes’, non-trackable Missiles, (such as Ballistic Missiles,) will be detonated instead of reaching the end target destination if diverted with Countermeasures.
- This is a customizable reworked behavior of Countermeasures (see below) in an attempt to make Countermeasures more realistic with less gameplay impact, as missiles diverted with Countermeasures no longer deal damage.

- New FPS System:

- As ’FramesPerSecondLimit’ does not change the in-game FPS, an expansion is added to enable customization of FPS features in the Main Menu and Single Player Games.

```
{ "NewSkirmishFPSSystem",    INI::parseBool, NULL, offsetof(GlobalData, m_newskirmishfpsSystem) }
{ "LoadSkirmishFPS",     INI::parseBool, NULL, offsetof(GlobalData, m_skirmishloadfps) },
{ "MenuScreenFPS",   INI::parseUnsignedInt, NULL, offsetof(GlobalData, m_menufps) },
{ "InGameFPS",   INI::parseUnsignedInt, NULL, offsetof(GlobalData, m_newfpsLimit) },
```

- Added NewSkirmishFPSSystem. If set to ‘Yes’, FPS won’t be set to Unlimited (Or 1000) when configured 61 Game Speed or above.
- Added InGameFPS. By Default, the FPS is locked at 30 for Single Player games. This enables the Default FPS to be overridden.
  NOTE: This also overrides the Menu Screen FPS.
- Added MenuScreenFPS. By default, the FPS of the Main Menu follows the ‘FramesPerSecondLimit’, this value overrides the Main Menu exclusively if the Shell Map (Animated Menu) is on.
- Added LoadSkirmishFPS. By Default, loading a Skirmish game from a Game Save does not register the Game Speed in the Skirmish option. Setting this to ‘Yes’ enables the FPS to be the same as the Game Speed.

NOTE:

> All of the New FPS Features fall under the ’FramesPerSecondLimit’. If they are configured higher, then the system will simply use ’FramesPerSecondLimit’.

> When configuring higher FPS, you are required to configure the Settings for the 2 files below in Windows big file for them to work properly.

```
NAME = "SkirmishGameOptionsMenu.wnd:SliderGameSpeed";

SLIDERDATA = MINVALUE: 15,
             MAXVALUE: 301;
```

```
NAME = "OptionsMenu.wnd:SliderScrollSpeed";

SLIDERDATA = MINVALUE: 1,
             MAXVALUE: 75;
```

- DrawFactor:

  - NOTE: Please use DrawEntireTerrain that is already present in the existing Source Code.

```
{ "DrawWidthFactor", INI::parseReal, NULL, offsetof( GlobalData, m_drawWidthFactor ) },
{ "DrawHeightFactor", INI::parseReal, NULL, offsetof( GlobalData, m_drawHeightFactor ) },
{ "DrawFullMap", INI::parseBool, NULL, offsetof( GlobalData, m_drawFullMap ) },
```

  - Added DrawWidthFactor, and DrawHeightFactor. You are able to adjust the DrawWidth and DrawHeight for compensating with higher CameraHeight values. The Draw Factor will never exceed the Map Values. Defaults to: 1.0. ~~After testing, 1.75-1.8 hits the sweet spot, 2.5 or higher is recommended if you have a good system.~~
    ~~NOTE: This is Quite Performance Taxing, recommend to only set it above 2.0 if you're on a Low End System.~~ Adjusted to fit DrawEntireTerrain sizes.

  - ~~Added DrawFullMap. Adjust the Draw Distance to be of the Full Map size. If this is set to 'Yes', ignore DrawWidthFactor, and DrawHeightFactor as they could not exceed the Map Width and Map Height values. Defaults to: No.~~
    ~~NOTE: This is Quite Performance Taxing. Recommended to only do it if you're not on a Low End System.~~

- Different Random Types:

```
{ "SeedRandomType",  INI::parseAsciiString, NULL, offsetof(GlobalData, m_initRandomType) },
```

  - Added SeedRandomType, enables different configuration for Random Logic. The current random logic is varying with every game load and is non-deterministic, but can still be predictable. This however can be changed with different configurations. Namely: "MORE_RANDOM", "EXHAUSTIVE", and "TIME". If the option is configured with anything other than the values mentioned, the game would use the default Random Value. NOTE: This Feature only Works in Single Player Games and might Break Playbacks!

  - MORE_RANDOM: Uses existing Seed Values to Randomize and Generate a New Seed. This enables more variation for seeding. It uses decimals but converts back to a whole number at the end of calculation. Formula below:

```
newSeed = GameLogicRandomValueReal(-PI,PI)*GameLogicRandomValue(0,100);
newSeed*= GameLogicRandomValueReal(0.0f,max(Real(GameLogicRandomValue(10,1e8)), Real(fabs(GetGameLogicRandomSeed()*GameLogicRandomValueReal(-newSeed, newSeed)))));
```

- EXHAUSTIVE: Modifies the formula for generating Random Factions and placing Players when the Player Slot is not assigned. The existing formula uses Modulo function and performs expensive operations to exhaust memory for finding a random slot. The function modifies the existing function with more variations, being more expensive than the existing function:

```
//
UnsignedInt silly = UnsignedInt((GetGameLogicRandomSeed()*GameLogicRandomValueReal(-2.0f,2.0f))) % 7;
Int verysilly = silly * GameLogicRandomValueReal(0.0f, Real(GetGameLogicRandomSeed() % 3));
silly = GameLogicRandomValue(0, verysilly);
for (UnsignedInt poo = 0; poo < silly; ++poo)
{
    GameLogicRandomValue(0, 1);   // ignore result
}
silly *= silly;
Int fullsilly = max(Int(silly+1), Int(1e10));
newSeed = GameLogicRandomValue(silly, fullsilly);
```

- TIME: Uses the current time as the seedling value. This is the least varying and the Most Deterministic random seeding. The seed would be EXACTLY the same if the game is loaded within a similar timeframe.

- iterateDrawablesInRegion Function Optimization:

- Added an attempted iterateDrawablesInRegion Function Optimization by using PartitionManager. Three Options can be configured in GameData.ini to use the configured functions.

```
{ "UsePartitionManagerToIterateDrawables",    INI::parseBool, NULL, offsetof(GlobalData, m_usePartit
{ "UsePartitionManagerToIterateDrawablesOnlySelect",      INI::parseBool, NULL, offsetof(GlobalData,
{ "UseEfficientIterateDrawablesScheme",   INI::parseBool, NULL, offsetof(GlobalData, m_useEfficientD
```

- Added UsePartitionManagerToIterateDrawables. This attempts to use PartitionManger to find Drawables to Iterate within the region. This affects both GameClient and W3DView. The current system iterates through all the drawables list and does the ones that are within the region. Setting this to ‘Yes’ uses this alternative drawing system. Defaults to: No.

  > GameClient uses the whole Screen onto World Coordinates as the radius for finding Drawables.W3DView uses the Selection Area onto World Coordinates as the radius for finding Drawables.

- Added UsePartitionManagerToIterateDrawablesOnlySelect. This limits the above function to only W3DView and not GameClient. Does nothing if

‘UsePartitionManagerToIterateDrawables’ is not configured with ‘Yes’.
Defaults to: No.

- Added UseEfficientIterateDrawablesScheme. When set to ‘Yes’, this applies a technical implementation for only GameClient that Registers the Drawables that are Iterated onto a List. The function is configured then only to Iterate through the Registered Drawables. ~~The PartitionManager Iterates through Borders of the Screen instead of the Whole Screen Region to find New Drawables that Fits onto the Screen and Registers them to the List.~~
  Defaults to: No.

  Other features about this technical implementation:

  > Registering a New Object or Drawable will add their Drawables onto the List if it is within the Screen Region.

  > Thing::setPosition also Registers the Object's Drawables onto the List if it is within the Screen Region.

  > PhysicsBehavior being Active will also add the Drawable onto the List if it is within the Screen Region.

  > Drawables that exit the Screen Region will be removed from the List.

  > Adjusting the Screen will also clear the Drawables List.

  > StealthUpdate that gives OBJECT_STATUS_DETECTED while the relationship is ENEMIES will clear the Drawables List.

  > ParticleCannon on changing its Firing Sequence will clear the Drawables List.

  > When the Drawables List is empty, the function uses the existing non-technical function to IterateDrawables.

  > Has a downside that may give nullptrs to Drawable::getDrawModules, but can be mitigated by checking to avoid crashes. This feature is currently implemented within the expansion.

  > NOTE: This may be the most efficient function, it may cause bugs in the Rendering since Drawables are not meant to iterate the way as this function intends.

  Extra features when Enabling this Feature:

  - W3DModelDraw: Turret Positioning and Recoil + Muzzle Handling won’t be updated every frame and will update only when necessary.

- Non-Retail AI Pathfind:

Variable below is OBSELETE, replaced with Non-Retail Compatible AIPathfind:

- ~~Added an attempted fix to make 8 player maps playable with more than 6 AIs. This should not be taken as a solution, but as a horrible workaround for making 8 player maps playable.~~

```
{ "AttemptToFixGroundLocomotorClump",    INI::parseBool, NULL, offsetof(GlobalData, m_fixLoco
```

- ~~Added AttemptToFixGroundLocomotorClump. If this is set to 'Yes' while there are 6 or more players present (including AI), the system will force any stationary Ground Objects to move if they remain stationary for the last half a second while they have been given orders to move.~~
- ~~Computer players will have the opposite effect and their movement is halted if they did not move from their position last checked.~~
- ~~This has been tested and does seem to fix some issues, but several of them still persist. (Such as playing with 7 Brutal AIs on Contra Mod.) Defaults to: No.~~

```
{ "UseNonRetailPathfindToFixPathfindForManyPlayers",     INI::parseBool, nullptr, offsetof(GlobalDat
{ "UseNonRetailAIPathfind",  INI::parseBool, nullptr, offsetof(GlobalData, m_useNonRetailAIPathfind)
{ "UseNonRetailAIPathfindAllocation",    INI::parseBool, nullptr, offsetof(GlobalData, m_useNonRetai
{ "UseNonRetailAIPathfindDynamicAlloc",  INI::parseBool, nullptr, offsetof(GlobalData, m_useNonRetai
{ "UseNonRetailAIPathfindOpenSortedList",    INI::parseBool, nullptr, offsetof(GlobalData, m_useNonR
```

- Functions above except 'UseNonRetailAIPathfindDynamicAlloc', and 'UseNonRetailAIPathfindOpenSortedList' are automatically Enabled when Compiling with RETAIL_COMPATIBLE_PATHFINDING Disabled.
- Added UseNonRetailPathfindToFixPathfindForManyPlayers. If this is set to 'Yes' while there are 6 or more players present (including AI), the system will use Non-Retail Pathfind System, along with 'NonRetailAIPathfindAllocation' (by TheSuperHackers) for the AIPathfinding. Defaults to: No.
- Added UseNonRetailAIPathfind. Setting this to 'Yes' configures the System to use Non-Retail Pathfind System (by TheSuperHackers), and with RETAIL_COMPATIBLE_PATHFINDING_ALLOCATION disabled. Defaults to: No.
- Added UseNonRetailAIPathfindAllocation. Setting this to 'Yes' configures the System to use Pathfind System with RETAIL_COMPATIBLE_PATHFINDING_ALLOCATION Disabled (by TheSuperHackers). Defaults to: No.

- Added UseNonRetailAIPathfindDynamicAlloc. Setting this to ‘Yes’ configures the System to use Dynamic Allocation Edits by Mauller for the Pathfind System. This feature is inherently Performance Expensive and not-recommended. Requires ‘UseNonRetailAIPathfind’ Set to ‘Yes’ for Enabling these features. Requires Defaults to: No.
- Added UseNonRetailAIPathfindOpenSortedList. An optimization attempt towards the Pathfind system by Mauller. Reports from SuperHacker devs show positive results towards Lategame. Requires ‘UseNonRetailAIPathfind’ Set to ‘Yes’ for Enabling these features. Defaults to: No.

- SlowDeathBehaviors Interactions:

```
{ "FlungCorpsesHasAirDrag", INI::parseBool, NULL, offsetof(GlobalData, m_corpsesHaveAirDrag) },
{ "FixHulksFreezingAboveTerrain",   INI::parseBool, NULL, offsetof(GlobalData, m_fixHulksFree
```

- Added FlungCorpsesHasAirDrag. Originally Corpses or Husks that are Flailing in the Air are being Disabled by DISABLED_HELD, disabling any External Forces from interacting with them. Setting this to ‘Yes’ enables external Forces to act upon them. Defaults to: No.
- Added FixHulksFreezingAboveTerrain. Husks or Debris that are in air while reaching their SinkDelay will result in the Husks or Debris frozen in air before Vanishing. Setting this to ‘Yes’ extends all three Delays of SlowDeathBehavior until it reaches the Ground. Defaults to: No.

- Display Money Interactions:

```
{ "HideCashFromShowingToEnemies",   INI::parseBool, NULL, offsetof(GlobalData, m_hideCash
{ "HideCashFromShowingToEnemiesInvisibleUnitsOnly", INI::parseBool, NULL, offsetof(Global
```

- Added HideCashFromShowingToEnemies. Cash earned by Allies, whether Supply, Hacking, or Pickups, can be seen by Enemies. Setting this to ‘Yes’ only allows Cash Earned to be seen if the relationship is either ALLIES or NEUTRAL. Defaults to: No.
- Added HideCashFromShowingToEnemiesInvisibleUnitsOnly. Same as above, but only Hide the Cash Earned if the Unit is either Stealthed or Disguised without being Detected. Setting this to ‘Yes’ overwrites the above Variable. Defaults to: No.

- Gameplay Geometrical Togglable Features:

```
{ "UseDynamicTargetingForWeapons",  INI::parseBool, NULL, offsetof(GlobalData, m_dynamicTargeting)
{ "UseAccurateSphereToRectCollision",   INI::parseBool, NULL, offsetof(GlobalData, m_useAccurateSp
{ "CheckBoxBoundariesForDistCalc",  INI::parseBool, NULL, offsetof(GlobalData, m_checkBoxBoundarie
```

- Added UseDynamicTargetingForWeapons. This is a targeting feature inspired by RA3, prominent in Gen: Evo mod. When attacking Objects, especially Structures, the Targeted Position will be a more Diverse, mimicking Realistic Modern Approach. The example is prominent here, especially with how the Bullets and Missiles target onto Vehicles, Aircraft and Structure.

  When enabled, Bullets, Lasers and Projectiles will target onto a random suitable Geometrical Information to imitate the RA3 mechanics. Defaults to: No.

- Added UseAccurateSphereToRectCollision. Adjusts the Collision detection between Spheres/Cylinders with Box to use a more Viable representation. The current formula sets the Sphere to Box Geometry and utilizes the Major Radius for Both the Minor and Major Radius.

  A more accurate formula is present in the source code but is not compatible with Rotation Checks. Setting this to ‘Yes’ utilizes an attempt to design the Collision feature based on the Existing Feature that works with Rotation Checks. Defaults to: No.

- Added CheckBoxBoundariesForDistCalc. An attempt to Design Workaround for Boundary Box Issue when Targeting Structures, which is able to Damage them Outside of Boundary Range. Setting this to ‘Yes’ enables it, Defaults to: No.

Commands:

- CommandButton:

> Reworks:

- Options - ALLOW_SHRUBBERY_TARGET: Reworked to be able to select previously Unselectable Shrubberies. However, the method does not actually Select the Shrubbery. It creates a Template at the detected location and the User uses the Created Template and its Position for carrying out the Command.

```
    "ENTER_ME",
    "EXECUTE_RAILED_TRANSPORT",
    "BEACON_DELETE",
    "SET_RALLY_POINT",
    "SELL",
    "FIRE_WEAPON",
    "SPECIAL_POWER",
    "PURCHASE_SCIENCE",
    "HACK_INTERNET",
    "TOGGLE_OVERCHARGE",
#ifdef ALLOW_SURRENDER
    "POW_RETURN_TO_PRISON",
#endif
    "COMBATDROP",
    "SWITCH_WEAPON",
    "HIJACK_VEHICLE",
    "CONVERT_TO_CARBOMB",
    "SABOTAGE_BUILDING",
    "EQUIP_OBJECT",
#ifdef ALLOW_SURRENDER
    "PICK_UP_PRISONER",
#endif
    "PLACE_BEACON",
    "SPECIAL_POWER_FROM_SHORTCUT",
    "SPECIAL_POWER_CONSTRUCT",
    "SPECIAL_POWER_CONSTRUCT_FROM_SHORTCUT",
    "SELECT_ALL_UNITS_OF_TYPE",

    "DISABLE_POWER",

    NULL
```

- New Command: DISABLE_POWER.

  > Manually disables the Powered Object or Power Plant. Disabling Powered Object this way disables them from consuming or generating power.

  > Requires Objects to be configured with POWERED or POWERED_TANK or being able to Generate Energy to utilize this command.

  > Units being Disabled with this Command are considered to be Disabled Underpowered.

- New Command: ENTER_ME.

> The opposite of EVACUATE command. Calls nearby Objects to Enter the Current Object, if able.

> Requires OrderNearyUnits properties to be configured for the command to function (See below.)

- New Command: EQUIP_OBJECT.

> Determines if the Target can satisfy the Objects EquipCrateCollide module, if so, when targeted, move the Object(s) to Equip the Targeted Object.

- New Guard Commands:

- New Command: GUARD_CURRENT_POS.

> Tell the Object to do Guard Stance at its current Position.

> Command is Non-Target Based. So it happens immediately after clicking the Command Button.

- New Command: GUARD_CURRENT_POS_WITHOUT_PURSUIT.

> Same as above, but configured in the way of 'GUARD_WITHOUT_PURSUIT'.

- New Command: GUARD_CURRENT_POS_FLYING_UNITS_ONLY.

> Same as above, but configured in the way of 'GUARD_FLYING_UNITS_ONLY'.

- New Command: GUARD_FAR.

> Tell the Object to do Guard Stance at the declared Position, but will do it from the Position Afar from the Center.

> Instead of going to the Center, the Object will Stay either at the Edge of the Guard Radius, or when its Current Weapon's Range can reach the Whole Covered Area.

> When declared if either of these two Conditions are met, the Object will stay at its Current Position to Guard the Area.

- New Command: GUARD_FAR_WITHOUT_PURSUIT.

> Same as above, but configured in the way of 'GUARD_WITHOUT_PURSUIT'.

- New Command: GUARD_FAR_FLYING_UNITS_ONLY.

> Same as above, but configured in the way of ‘GUARD_FLYING_UNITS_ONLY’.

- NEW Order Nearby Units:

```
{ "OrderNearbyUnitsRadius",             INI::parseReal, NULL, offsetof( CommandButton,
{ "OrderNearbyUnitsKindof",                 KindOfMaskType::parseFromINI,       NULL,
{ "OrderNearbyUnitsForbiddenKindof",        KindOfMaskType::parseFromINI,       NULL,
{ "OrderNearbyUnitsMinDelay",       INI::parseDurationUnsignedInt,      NULL, offsetof
{ "OrderNearbyUnitsMaxDelay",       INI::parseDurationUnsignedInt,      NULL, offsetof
{ "OrderNearbyUnitsIntervalDelay",  INI::parseDurationUnsignedInt,      NULL, offsetof
```

- Added OrderNearbyUnitsRadius. When configured, commands nearby units within the range under the declared Radius to do the same Command as the Command declared under the CommandButton (Except for ENTER_ME, See Above.) Only units that are valid for the Command will carry out the Command, others that are invalid will not be doing the Command.
  Defaults to: 0, or None.
- Added OrderNearbyUnitsKindof. Determines the Multiple Required Kindofs needed to present for a unit to be able to carry out the Command from OrderNearbyUnitsRadius.
  Defaults to: None, or all are able to carry out the Command.
- Added OrderNearbyUnitsForbiddenKindof. Determines Any Kindofs present within a unit to Not be able to carry out the Command from OrderNearbyUnitsRadius.
  Defaults to: None.
- Added OrderNearbyUnitsMinDelay. Determines the minimum delay for the Unit to carry out the Command from OrderNearbyUnitsRadius. Manually Commanding the Units before the Delayed Command is Carried out prevents the Delayed Command from being Carried out.
  Defaults to: 0, or None.
- Added OrderNearbyUnitsMaxDelay. Determines the maximum delay for the Unit to carry out the Command from OrderNearbyUnitsRadius. To configure a constant Delay, set the same value for both ‘OrderNearbyUnitsMinDelay’ and ‘OrderNearbyUnitsMaxDelay’.
  Defaults to: 0, or None.
- Added OrderNearbyUnitsIntervalDelay. Set the Interval Delay between each unit called from OrderNearbyUnitsRadius to carry out their Command. Rules also apply the same to the above, manually Commanding the Units before the

Delayed Command is Carried out prevents the Delayed Command from being Carried out.
Defaults to: 0, or None.

- NEW Custom Radius Cursors

```
{ "RadiusCursorType",          INI::parseIndexList,              TheRadiusCursorNames, offset
{ "CustomRadiusCursorType",        INI::parseAsciiString,          NULL, offsetof( CommandButton
```

- Added CustomRadiusCursorType. If configured, bypasses RadiusCursorType and uses the CustomRadiusCursor with the name declared outside of the InGameUI (See above CustomRadiusCursor). Defaults to: Not configured.

- Formation Mode:

> There is an existing Movement Mode that is configured when Pressing CTRL + F onto Selected Objects. configuring them to Move in Formation Mode.

> If all the Selected Objects are configured to Move in formation Mode from the Same Selection, they will Move while Preserving their Coordinates from the “Center Position” of Formation.

> Units moved in Formation have slower and restricted Movement Speed and Break Formation if an Airborne Unit is included in the Formation.

> Formation Mode has been reworked as follows:

- Moving Airborne Units in Formation no longer Breaks the Formation Mode.
- Better Airborne Units Positioning while in Formation Mode (Still bad.)

- Command Map:

```
{ "Transition",                INI::parseLookupList,                          Transit
```

- Restored DOUBLEDOWN Feature for Transition. Previously this feature is Disabled as it did not function correctly. This has been redesigned and restored.
- New CommandMap: MOVE_IN_FORMATION:

  > When triggered, enables a Movement Mode that tells Objects to Move in Formation without the need to register Objects for Formation.

  > When telling Units to Move, they are registered with a new Formation Mode.

  > If all the Selected Objects have Previously Moved is this Mode within the same Group, it will use the Last “Center” from the Formation Mode.

  > Units that Move in this Mode for Moving Command do not have a Movement Speed Penalty.

> An Object that has been registered with Formation Mode will be separated to use its Formation Center and Movement Speed Limit.

> While this Mode is On, the Mouse will be switched to ‘MoveInFormation’ Mouse Cursor (This is required to be declared in Mouse.ini.)

> The Mouse Cursor can be customized with ‘MoveInFormationCursorName’ on the Player Template, it will still prioritize ‘MoveCursorName’ for Objects if present.

Special Powers:

```
{ "DeleteUserOnExecute",                INI::parseBool,                 NULL,
{ "FXOnExecute",                        INI::parseFXList,                   NU
{ "OCLOnExecute",                       INI::parseObjectCreationList,       NU
```

- Added DeleteUserOnExecute. If set to ‘Yes’, it will delete the object after using the declared SpecialPowerTemplate.
- It uses the same form of Deletion used by DeletionUpdate, which would not inform or alert the system on delete.
- This is preferred for Superpowers or Special Powers that are executable from Control Bar, as Special Powers for units are often triggered when executing abilities before completing the effects. For units, check out SpecialAbilityUpdate.
- Added FXOnExecute and OCLOnExecute. If set, spawns the respective FX and OCL on the Object after using the SpecialPowerTemplate.

System:

- Disabled Types:

```
  "DEFAULT",
  "DISABLED_HACKED",
  "DISABLED_EMP",
  "DISABLED_HELD",
  "DISABLED_PARALYZED",
  "DISABLED_UNMANNED",
  "DISABLED_UNDERPOWERED",
  "DISABLED_FREEFALL",

"DISABLED_AWESTRUCK",
"DISABLED_BRAINWASHED",
  "DISABLED_SUBDUED",

  "DISABLED_SCRIPT_DISABLED",
  "DISABLED_SCRIPT_UNDERPOWERED",

  "DISABLED_FROZEN",
  "DISABLED_STUNNED",

  "DISABLED_TELEPORT",
  "DISABLED_CHRONO",
```

- Added two more DisabledTypes: DISABLED_FROZEN and DISABLED_STUNNED:

  > DISABLED_FROZEN: Same features as DISABLED_SUBDUED, except it doesn't have a Tint, Poweroff Icon appearing on top of the unit, and does not play Power On/Off sound when set or removed.

  > DISABLED_STUNNED: Same features as DISABLED_PARALYZED, except it doesn't have a Tint and Poweroff Icon appear on top of the unit. DISABLED_PARALYZED, unlike DISABLED_SUBDUED, doesn't stop Containing features such as Garrison, Entering/Exiting and Open-Topped.

- Status Types:

- Reworked Status Types:

  > IMMOBILE: Feature has been reworked. No longer just disables the Object from actively moving, now also disables their Locomotor, preventing them from being able to move and disables them from retaliating. The difference being they are able to 'remember' their last movement destination after removing the status. They are also considered as a type of pseudo-structure, as they can block buildings and movement pathways. AI behaviors are also updated, they do not aim outside of their attack range, are considered stationary for AI pathfinding, and AI teams do not consider them for leadership.

Current implementation for using this feature is the Reworked DeployStyleAIUpdate's 'DeployNoRelocate' feature, which not only Disables the Object from moving, but also considers it to be an Obstacle that hinders Building Deployment.

EMPUpdate:

- Expanded EMPUpdate to be able to work with Projectiles. Projectiles affected by EMPUpdate are now killed instantly. You can set the death Type for Projectiles or configure them to be Jammed.

```
{ "DoesNotAffect", INI::parseBitString32,   TheWeaponAffectsMaskNames, offsetof(EMPUpdateModuleData, m_re
{ "DoesNotAffectMyOwnBuildings", INI::parseBool, NULL, offsetof( EMPUpdateModuleData, m_doesNotAffectMyOw
{ "DisabledType",    DisabledMaskType::parseSingleBitFromINI, NULL, offsetof( EMPUpdateModuleData, m_disab
{ "AffectsKindOf", KindOfMaskType::parseFromINI, NULL, offsetof( EMPUpdateModuleData, m_affectsKindOf ) }
{ "TintStatusType",         TintStatusFlags::parseSingleBitFromINI, NULL, offsetof( EMPUpdateModuleData,
{ "CustomTintStatusType",       INI::parseAsciiString,  NULL, offsetof( EMPUpdateModuleData, m_customTint

{ "EMPProjectileJammed",    INI::parseBool, NULL, offsetof( EMPUpdateModuleData, m_empProjectileSubdual)
{ "EMPProjectileDeathType", INI::parseAsciiString, NULL, offsetof( EMPUpdateModuleData, m_empProjectileDe
{ "EMPProjectileCustomDeathType", INI::parseQuotedAsciiString, NULL, offsetof( EMPUpdateModuleData, m_emp
```

- Reworked DoesNotAffect. Now works for not just ALLIES, but also ENEMIES and NEUTRALS. Defaults to: None.
- Added DisabledType. You can now customize the Disable type for the EMPUpdate. Defaults to: DISABLED_EMP.
- Added AffectsKindOf. You can now customize the KindOfs EMP is able to affect, including Projectiles. The function searches for Any KindOfs declared. Defaults to: VEHICLES, STRUCTURES, AIRCRAFT, and SPAWNS_ARE_THE_WEAPONS.
- Added TintStatusType. Customizes the TintStatus for the EMP Duration. Can be used to override default Tint for Disabled Types. Defaults to: TINT_STATUS_DISABLED.
- Added CustomTintStatusTypes. For compatibility with CustomTintStatuses. Defaults to: None.
- Added EMPProjectileJammed. If set to 'Yes', projectiles affected by the EMP are considered Jammed as similar to ECM deflection. Defaults to: No.
- Added EMPProjectileDeathType. EMPed Projectiles now instantly dies. This configures the DeathType for EMPed Projectiles. Defaults to: LASERED. If EMPProjectileSubdual is configured 'Yes', this feature will be overridden.
- Added EMPProjectileCustomDeathType to support CustomDeathType. Defaults to: None.

LeafletDropBehavior:

- Module fused with EMPUpdate. You can now parse EMPUpdate variables on LeafletDropBehavior.
- However, you are still required to define “AffectRadius” for the Effect Radius for LeafletDropBehavior. Defaults to: 60 units.
- DoesNotAffect for LeafletDropBehavior defaults to: ALLIES, and NEUTRALS.
- AffectsKindOf for LeafletDropBehavior defaults to: INFANTRY, and VEHICLES.

Projectiles:

- Expanded Subdual types to work with other Projectile types and not just Missiles. Projectiles with DumbProjectileBehavior, FreeFallProjectileBehavior, or NeutronMissileUpdate can now be Subdued.

```
{ "AllowSubdual", INI::parseBool, NULL, offsetof(DumbProjectileBehaviorModuleData, m_allowSubdual) }
{ "AllowAttract", INI::parseBool, NULL, offsetof(DumbProjectileBehaviorModuleData, m_allowAttract) }
{ "DistanceScatterWhenJammed",INI::parseReal,       NULL, offsetof(DumbProjectileBehaviorModuleData,
```

- Added AllowSubdual and AllowAttract. Since Projectile Subdual now consists of two types, (See SubdualIsMissileAttractor,) added options to configure if the Projectile allows types of Subduals.
- AllowSubdual is the default behavior for SUBDUAL_MISSILE. It will jam the Projectile, leaving them scattered from their intended position. Defaults to: Yes.
- AllowAttract will allow Weapons that uses SubdualMissileAttractor to redirect it to the Attractor. Defaults to: Yes.
- NOTE: For all types of SUBDUAL, Projectiles are required to have configured SubdualDamageCap higher than their Health in order for Subduals to work.
- Added DistanceScatterWhenJammed. Functions exactly the same as the configuration for MissileAIUpdate but now works on DumbProjectileBehavior, FreeFallProjectileBehavior, and NeutronMissileUpdate.

Weapons:

- Weapon Expanded Features:

- More memory allocated to WeaponTemplate {360, 32} to {512, 32}.

```
// New Features
{ "CustomDamageType",                    INI::parseAsciiString,  NULL,
{ "CustomDamageStatusType",              INI::parseAsciiString,  NULL,
{ "CustomDeathType",                     INI::parseAsciiString,  NULL,
```

- Added CustomDamageType for the weapon, with its features stated above.
- Added CustomDamageStatusType for the weapon. Overrides DamageStatusType if declared.
- Added CustomDeathType for the weapon to trigger. Overrides DeathType if configured.

- DamageTypes Based Features:

```
{ "WeaponIsFlame",                       INI::parseBool,         NULL,
{ "WeaponFlameCollidesWithBurning",      INI::parseBool,         NULL,
{ "WeaponIsPoison",                      INI::parseBool,         NULL,
{ "WeaponPoisonMuzzleFlashesGarrison",   INI::parseBool,         NULL,
{ "WeaponIsDisarm",                      INI::parseBool,         NULL,
{ "WeaponKillsGarrison",                 INI::parseBool,         NULL,
{ "WeaponKillsGarrisonAmount",           INI::parseInt,          NULL,
```

- Added WeaponIsFlame which considers whether the weapon is considered a FLAME weapon without DamageType = FLAME. Setting yes allows a weapon to set objects on Fire and allows their own Projectiles to pass through burning targets.
- Added WeaponFlameCollidesWithBurning. Setting yes disables the Projectile from passing through Burning targets. Applies to FLAME, and PARTICLE_BEAM DamageTypes, and Weapons with WeaponIsFlame set to yes.
- Added WeaponIsPoison which considers whether the weapon is considered a POISON weapon without DamageType = POISON. Setting to yes causes the weapon to trigger Object's PoisonedBehavior as with POISON DamageTypes. They also have no MuzzleFlashes when firing within Garrison.
- Added WeaponPoisonMuzzleFlashesGarrison. Setting yes enables Garrisoned units to show MuzzleFlashes when firing a weapon with POISON DamageType or with WeaponIsPoison set to yes.

- Added WeaponIsDisarm which considers whether the weapon is considered a DISARM weapon without DamageType = DISARM. Setting yes allows the weapon to clear mines as with DISARM weapons. Skips if the weapon DamageType is already DISARM.
- Added WeaponKillsGarrison which determines whether a weapon considers its Damage to Kill Garrisons without DamageType = KILLS_GARRISONED.
- Added WeaponKillsGarrisonAmount which overrides the Amount of Kill Garrisons for WeaponKillsGarrison.

- Targeting or Affects Based Features:

```
{ "RadiusDamageAffects",              INI::parseBitString32,  TheWeaponAffectsI
```

- Adjusted RadiusDamageAffects. Can now declare RadiusDamageAffects = SELF independently from ALLIES. Previously skips SELF checks if ALLIES is not declared. This is useful if you want a weapon with ProjectileObject to affect only yourself.
- If declared on a weapon that does not use 'ProjectileObject', the weapon will affect both the Target and the User. For a weapon that doesn't use a Projectile, see below 'DamagesSelfOnly'.

```
{ "DamagesSelfOnly",                          INI::parseBool,         NULL,
```

- Added DamagesSelfOnly. Weapons without Projectiles can now be used to only affect the user. If this is configured, weapons without projectiles can only damage or affect the user and not the target. Defaults to: No.
- When used on Weapons with 'ProjectileObject', the weapon will deal no damage because the system considers the Projectile independent from the User.

```
{ "WeaponUseSpecificVoice",           INI::parseAsciiString,  NULL,
{ "OverrideDamageTypeFX",             DamageTypeFlags::parseSingleBitFromINI,
```

- Added WeaponUseSpecificVoice which overrides the VoiceName when trying to use the Weapon on Attack or on the Command Bar. Defaults to: None.
- Added OverrideDamageTypeFX which overrides the DamageTypeFX when configured to anything other than UNRESISTABLE. Defaults to: UNRESISTABLE.

```
{ "MinDamageAltitude",                    INI::parseReal,                 NULL,
{ "MaxDamageAltitude",                    INI::parseReal,                 NULL,
{ "MinTargetAltitude",                    INI::parseReal,                 NULL,
{ "MaxTargetAltitude",                    INI::parseReal,                 NULL,
```

- Added mechanics for Weapons that calculates an Object's altitude before being able to Target or deal damage to it. Structures do not have altitude, but use their GeometryHeight for the calculation.
- Added MinDamageAltitude, which determines the minimum Height of the Unit required to be above the Surface for the weapon to be able to deal damage to it. Defaults to: 0 or at Ground.
- Added MaxDamageAltitude, which determines the maximum Height of the Unit required to be above the Surface for the weapon to be able to deal damage to it. Structures will always pass this criteria. Defaults to: 0 or no Limit.
- Added MinTargetAltitude, which determines the minimum Height of the Unit required to be above the Surface for the weapon to be able to target it. Defaults to: 0 or at Ground.
- Added MaxTargetAltitude, which determines the maximum Height of the Unit required to be above the Surface for the weapon to be able to target it. Structures will always pass this criteria. Defaults to: 0 or no Limit.

```
{ "PriorityWithinAttackRange",            INI::parseInt,                  NULL,
{ "PriorityOutsideAttackRange",           INI::parseInt,                  NULL,
```

- Added a priority system which determines the weapon to use against the Target Position or Object if it is within the Weapon's Attack Range or Outside or it.
- The unit will use the Weapon with the Highest Priority Number instead of the weapon that deals the Highest Damage or has the Longest Range for the WeaponChoiceCriteria.
- Priorities can be assigned with any number, but at the end, only the weapon with the Highest number will be chosen.
- If a Priority with number 1 or higher is assigned, the Unit's Weapon Slot will be Locked while Pursuing or Attacking until it is issued another command.
- Added PriorityWithinAttackRange, which determines the priority for the weapon when the Target is within the Weapon's Attack Range. Defaults to: 0 or none.
- Added PriorityOutsideAttackRange, which determines the priority for the weapon when the Target is outside of the Weapon's Attack Range. Defaults to: 0 or none.

```
{ "InvulnerabilityDuration",                    INI::parseDurationUnsignedInt,
```

- Added InvulnerabilityDuration. After hitting an Object, configure the Duration for Objects to be Always Considered as ALLIES for all Objects and Players. Uses the same configuration mechanic as ‘InvulnerableTime’ present for OCLs. This uses the same Mechanic as Undetected Defector. Does not actually make the Object hit with the Weapon Invulnerable for the Duration, but applies Undetected Defector onto the Object(s) for the duration. Defaults to: 0, or not configured.

- DamageStatusType Features:

```
{ "DamageStatusType",                   ObjectStatusMaskType::parseSingleBitFromINI,
```

```
{ "WeaponDoStatusDamageType",             parseAllVetLevelsBool,          NULL,
{ "VeterancyWeaponDoStatusDamageType",    parsePerVetLevelBool,           NULL,
{ "WeaponStatusDuration",                 INI::parseReal,         NULL,
{ "WeaponStatusDurationDamageCorrelation", INI::parseBool,        NULL,
{ "WeaponStatusTintStatus",               parseAllVetLevelsTintStatus,              NULL,
{ "VeterancyWeaponStatusTintStatus",      parsePerVetLevelTintStatus,               NULL,
{ "WeaponStatusCustomTintStatus",         parseAllVetLevelsAsciiString,             NULL,
{ "VeterancyWeaponStatusCustomTintStatus", parsePerVetLevelAsciiString,             NULL,
```

- DamageStatusType System has been reworked. It now indicates whether a StatusType has been removed from the Object (whether by the Object's actions or by Internal Code.)
- DamageStatusTypes can now be stacked on top of each other. Each have their own independent time to clear instead of only one being applicable at a time.
- Added WeaponDoStatusDamageType that enables the weapon to apply DamageStatus or CustomDamageStatus while not requiring DamageType = STATUS declared.
- Added VeterancyWeaponDoStatusDamageType, which allows the customization of WeaponDoStatusDamageType for Veterancy Levels.
- Added WeaponStatusDuration which overrides the duration of the DamageStatus applied instead of using the damage amount for calculating the Status Duration.
- Added WeaponStatusDurationDamageCorrelation which correlates the WeaponStatusDuration to their DamageType, calculating the Status Duration according to target’s armor.
- Added WeaponStatusTintStatus and WeaponStatusCustomTintStatus for the DamageStatus. It inflicts the Object with the declared TintStatus or CustomTintStatus.

- Added VeterancyWeaponStatusTintStatus and VeterancyWeaponStatusCustomTintStatus, which allows the customization of WeaponStatusTintStatus and WeaponStatusCustomTintStatus for Veterancy Levels.

- Firing Tracker Features:

```
{ "FiringTrackerStatusTrigger",             ObjectStatusMaskType::parseSingleBitFromINIVector,
{ "FiringTrackerBonusConditionType",        INI::parseIndexList,    TheWeaponBonusNames,
{ "FiringTrackerCustomStatusTrigger",               INI::parseAsciiStringVector,    NULL,
{ "FiringTrackerCustomBonusConditionType",          INI::parseQuotedAsciiString,    NULL,
```

- Added Mechanics that work like Avenger's Designator. It works when targeting a target that is affected by a StatusType. If the target has a StatusType or one of the StatusTypes declared, it grants the attacking Unit a WeaponBonus condition.
- Added FiringTrackerStatusTrigger which indicates what StatusType required for the target to grant the WeaponBonusCondition to the unit.
- Added FiringTrackerBonusConditionType which indicates the WeaponBonusConditionType to grant if the target has the required FiringTrackerStatusTrigger.
- Added functionality towards CustomStatusTypes and CustomWeaponBonusConditionTypes.
- These two do not overlap with each other and you can grant one Weapon each a WeaponBonusCondition and CustomWeaponBonusCondition.

For Example,

Weapon CrusaderTankGun

CustomWeaponBonus = CUSTOMBONUS1 RATE_OF_FIRE 150%

CustomWeaponBonus = CUSTOMBONUS1 RANGE 150%

FiringTrackerStatusTrigger = UNDER_CONSTRUCTION FAERIE_FIRE

FiringTrackerCustomStatusTrigger = HIGHLIGHT CUSTOMSTATUS1

FiringTrackerCustomBonusConditionType = CUSTOMBONUS1

End

When a unit firing CrusaderTankGun is attacking an object that is Under Construction, Highlighted by an Avenger or with the HIGHLIGHT, or

CUSTOMSTATUS1 CustomStatusType, it will grant the unit CUSTOMBONUS1. And while the unit is using CrusaderTankGun, it will receive bonuses declared above.

- NEW Requirement Features:

```
{ "TriggeredBy",                    INI::parseAsciiStringVector,    NULL,
{ "ConflictsWith",                  INI::parseAsciiStringVector,    NULL,
{ "RequiresAllTriggers",            INI::parseBool,                 NULL,
{ "RequiredStatus",                 ObjectStatusMaskType::parseFromINI, NULL,
{ "ForbiddenStatus",                ObjectStatusMaskType::parseFromINI, NULL,
{ "RequiredCustomStatus",           INI::parseAsciiStringVector,    NULL,
{ "ForbiddenCustomStatus",          INI::parseAsciiStringVector,    NULL,
```

- Added TriggeredBy, ConflictsWith, and RequiresAllTriggers. They are assigned the same way as Upgrade Modules. If assigned, they check whether a Weapon meets the Upgrade criteria, and if not, disable the User from using the Weapon. Defaults to: None.
- Added RequiredStatus, and ForbiddenStatus. They are assigned the same way as Behavior Modules. If assigned, they check whether a Weapon meets the Status criteria, and if not, disable the User from using the Weapon. Defaults to: None.
  NOTE: RequiredStatus checks for All of the declared Statuses while ForbiddenStatus checks for Any of the declared Statuses.
- Added RequiredCustomStatus, and ForbiddenCustomStatus to support CustomStatusTypes.

- NEW Cursor Features:

```
{ "AttackCursorName",               INI::parseAsciiString,      NULL,
{ "ForceAttackObjectCursorName",    INI::parseAsciiString,      NULL,
{ "ForceAttackGroundCursorName",    INI::parseAsciiString,      NULL,
{ "InvalidCursorName",              INI::parseAsciiString,      NULL,
```

- Added AttackCursorName. Changes the Attack Cursor when Able to Target for attacks if configured. Defaults to: Not configured.
- Added ForceAttackObjectCursorName. Changes the Force Attack Cursor when Able to Target for attacks if configured. Defaults to: Not configured.
- Added ForceAttackGroundCursorName. Changes the Force Attack Cursor when Targeting Ground for attacks if configured. Defaults to: Not configured.
- Added InvalidCursorName. Changes the Invalid Cursor when Unable to Target for attacks if configured. Defaults to: Not configured.

- Shockwave Features:

```
{ "ShockWaveUseCenter",                INI::parseBool,          NULL,
{ "ShockWaveRespectsCenter",           INI::parseBool,          NULL,
{ "ShockWaveAffectsAirborne",          INI::parseBool,          NULL,
{ "ShockWavePullsAirborne",            INI::parseBool,          NULL,
```

- ShockWave has been reconfigured to support Negative values. Negative values pull targets towards the center instead of pushing targets away from it.
- Added ShockWaveUseCenter. ShockWave uses the position of the Attacker and Target to determine the Positional Vector. Setting this to yes will use the Target Position for Vector calculation instead. Defaults to: No.
- Added ShockWaveRespectsCenter. Works like ShockWaveTaperOff but inverse. Determines how far the Target is from the Center and if it is less than the ShockWaveAmount, then it reconfigures the Amount to how far it is from the Center. Defaults to: No.
- Added ShockWaveAffectsAirborne. Shockwave normally does not affect Airborne units, this configuration allows it to be overridden. Defaults to: No
- Added ShockWavePullsAirborne. Shockwave pulls Airborne Units to the ground instead of pushing them higher into the air. Defaults to: No.

- NEW Magnet Force features:

```
{ "MagnetAmount",                      INI::parseReal,          NULL,
{ "MagnetInfantryAmount",              INI::parseReal,          NULL,
{ "MagnetTaperOffDistance",            INI::parseReal,          NULL,
{ "MagnetTaperOffRatio",               INI::parseReal,          NULL,
{ "MagnetTaperOnDistance",             INI::parseReal,          NULL,
{ "MagnetTaperOnRatio",                INI::parseReal,          NULL,
```

- Added a new mechanic, Magnet Force. Magnet Force works like Magnetron from RA2, able to pull units towards the Attacker. The mechanics have been extended and customized. The Force can be determined to be Push or Pull, and it can be used as an AOE ability. Only Objects with Locomotors are able to be affected by Magnet Force.
- Added MagnetAmount. Determines the Amount of Force applied. Setting positive values will tell the Magnet Force to apply Pushing force, while setting negative values will tell the Magnet Force to apply Pulling force. Defaults to: 0.
- Added MagnetInfantryAmount. Infantries are very susceptible to Magnet Force. MagnetInfantryAmount sets a separate Value exclusively for Infantry. If not configured, Magnet Force will use MagnetAmount for its calculation. Defaults: 0.

- Added MagnetTaperOffDistance. Works like ShockWaveTaperOffDistance, the further the Target is from the Attacker from the Determined Distance, the lower the Magnet Force. Defaults to: 0 or not applied.
- Added MagnetTaperOffRatio. Determines the rate of the effect of MagnetTaperOffDistance based on how far the Target is from the Attacker from the Determined Distance. Can be configured with Negative values for reverse effect. Defaults to: 1.0.
- Added MagnetTaperOnDistance. The inverse of MagnetTaperOffDistance, the closer the Target is from the Attacker from the Determined Distance, the higher the Magnet Force. Defaults to: 0 or not applied.
- Added MagnetTaperOnRatio. Determines the rate of the effect of MagnetTaperOnDistance based on how near the Target is from the Attacker from the Determined Distance. Can be configured with Negative values for reverse effect. Defaults to: 1.0.

```
{ "MagnetLiftHeight",                    INI::parseReal,                NULL,
{ "MagnetLiftHeightSecond",              INI::parseReal,                NULL,
{ "MagnetLiftForce",                     INI::parseReal,                NULL,
{ "MagnetLiftForceToHeight",             INI::parseReal,                NULL,
{ "MagnetLiftForceToHeightSecond",       INI::parseReal,                NULL,
{ "MagnetMaxLiftHeight",                 INI::parseReal,                NULL,
{ "MagnetLevitationHeight",              INI::parseReal,                NULL,
```

- Added MagnetLiftForce. Determines how much Lift Force the Magnet Force applies to Units. Ground Units are required to have a Lift Force for optimal effect. NOTE: MagnetLiftForce does not Affect Airborne Units. Defaults to: 1.0.
- Added MagnetLiftHeight. Determines if a different Lift Force should be used if the Target has not reached a certain amount of Height from the Surface. Defaults to: 0 or none.
- Added MagnetLiftForceToHeight. Determines the Amount of LiftForce to use for MagnetLiftHeight. Defaults to: 1.0.
- Added MagnetLiftHeightSecond. Determines if a different Lift Force should be used has not reached another certain amount of Height from the Surface. MagnetLiftHeightSecond is required to be configured higher than MagnetLiftHeight to work. Defaults to: 0 or none.
- Added MagnetLiftForceToHeightSecond. Determines the Amount of LiftForce to use for MagnetLiftHeightSecond. Defaults to: 1.0.

- Added MagnetMaxLiftHeight. Determines the maximum Height of the Units for the Object to apply Lift Force. After reaching the maximum height, the Lift Force will be significantly reduced to not being impactful. Defaults to: 0 (or None if not configured.)
- Added MagnetLevitationHeight. If the Object affected by Magnet Force is above the configured Height from the Surface, the Object will constantly stay above the Height. If the Object is not constantly attacked by a Magnet Force with the same MagnetLevitationHeight, the Object will be cleared from Levitation. After clearing from Levitation, the Attacker will be cleared from using its weapon. NOTE: This does not affect Air Units. Defaults to: 0 or none.

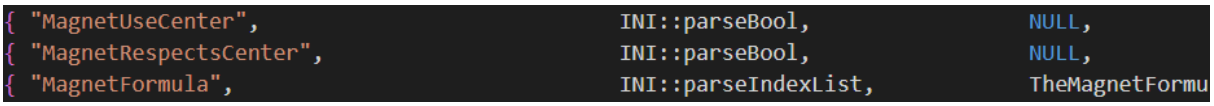
```
{ "MagnetMinDistance",                  INI::parseReal,                 NULL,
{ "MagnetMaxDistance",                  INI::parseReal,                 NULL,
{ "MagnetLinearDistanceCalc",           INI::parseBool,                 NULL,
```

- Added MagnetMinDistance. Configure the Minimum Distance the Target is to the Source required to trigger Magnet Force. Units within the minimum distance cannot be affected by the Weapon's Magnet Force. Defaults to: 0, or none.
- Added MagnetMaxDistance. Configure the Maximum Distance the Target is to the Source to be able to trigger Magnet Force.Units outside the maximum distance cannot be affected by the Weapon's Magnet Force. Defaults to: 0 (or Unlimited when not configured).
- Added MagnetLinearDistanceCalc. Configure whether the MagnetMinDistance and MagnetMaxDistance uses Linear2D distance instead of 3D distance. Defaults to: No.

```
{ "MagnetNoAirborne",                   INI::parseBool,                 NULL,
{ "MagnetAirborneZForce",               INI::parseReal,                 NULL,
{ "MagnetAirboneAffectedByYaw",         INI::parseBool,                 NULL,
```

- Added MagnetNoAirborne. As Magnet Force can be used against Airborne Units, this can disable it. Defaults to: No.
- Added MagnetAirborneZForce. By default, the system will use the processed Magnet Force for the ZForce to push or pull Airborne Units higher or lower. This enables the ZForce for the Magnet Force to be Overridden towards Airborne Units. Defaults to: 0 or use processed Magnet Force.
- Added MagnetAirborneAffectedByYaw. The application of Magnet Force applies Acceleration according to the Magnet Force's direction. However, the Yaw from Airborne Units does not properly render the mechanic. The default Airborne

Magnet Mechanic modifies the Magnet Force to use Velocity values for a more accurate representation but in turn Jams their movement. This changes the Magnet Force to apply Acceleration, but in turn makes the Magnet Force to have a different representation. Defaults to: No.



- Added MagnetUseCenter. Determines the direction to apply Magnet Force should be based on the Target's Position rather than the Position of the Attacker. Defaults to: No.
- Added MagnetRespectsCenter. Determines if the MagnetAmount of Magnet Force should scale lower based on how close the Target is from the Determined Position. Defaults to: Yes.
- Added MagnetFormula. Can configure 4 types of Magnets to use for applying Magnet Force, 'STATIC', 'DYNAMIC', 'ROTATORY', and 'HYPERDYNAMIC'. Can only be configured with one Magnet Formula. Defaults to: STATIC.
- 'STATIC' removes the Unit's previous velocity and uses a set Force for scaling the Magnet Force.
- 'DYNAMIC' removes the Unit's previous velocity and incorporates the Unit's Mass into the Magnet Force. The lighter the Unit, the Higher the Efficiency.
- 'ROTATORY' adds the previous Unit's velocity and uses a set Force for scaling the Magnet Force.
- 'HYPERDYNAMIC' adds the previous Unit's previous velocity and incorporates the Unit's Mass into the Magnet Force. The lighter the Unit, the Higher the Efficiency.
- 'DYNAMIC' and 'HYPERDYNAMIC' are less efficient than their 'STATIC' or 'ROTATORY' counterparts as a Unit's Mass is more likely to be considered Heavy.
- 'STATIC' and 'DYNAMIC' are better suited for applying constant Magnetic Forces and Levitation mechanics while 'ROTATORY', and 'HYPERDYNAMIC' are better suited for one-time weapons such as Magnet Bombs.

- NEW Railgun Features:

```
{ "IsRailgun",                    INI::parseBool,              NULL,
{ "RailgunIsLinear",              INI::parseBool,              NULL,
{ "RailgunUsesSecondaryDamage",   INI::parseBool,              NULL,
{ "RailgunPiercesBehind",         INI::parseBool,              NULL,
{ "RailgunPierceAmount",          INI::parseInt,               NULL,
```

- Added IsRailgun. If Set to ‘Yes’ for Weapons with No ProjectileObject, configure to damage the Objects encountered in-between the User and the Target or Position. Railguns only damage Objects that are between the User and the Target or Position. The Primary Target, if any, cannot be damaged by the Railgun. Does nothing for Weapons with No ProjectileObject. Defaults to: No.
- Added RailgunIsLinear. Sets whether the Weapon uses the User’s Firing Offset to adjust the Firing Position. If Set to ‘No’, the Railgun starts from the User's standing position. Railguns are configured to scan from the current Firing Height or Ground Height, if it is higher. Defaults to: Yes.
- Added RailgunUsesSecondaryDamage. Configure whether Railgun uses SecondaryDamage for Damage calculation. By default, Railgun uses PrimaryDamage for Damage calculation. Defaults to: No.
- Added RailgunPiercesBehind. Railgun uses a Range-based Iterate check with Radius to see whether current targets can be damaged. Originally, this includes checking if the current target is behind the Primary Target or Position. Setting this to ‘Yes’, allows targets that are behind the Primary Target or Position to be damaged. Defaults to: No.
- Added RailgunPierceAmount. Configure the Maximum Amount of Objects that can be damaged by the Railgun, starting from the Nearest Targets to the Furthest. Defaults to: -1 or Unlimited.

```
{ "RailgunRadius",                INI::parseReal,              NULL,
{ "RailgunRadiusCheckPerDistance",INI::parseReal,              NULL,
{ "RailgunInfantryRadius",        INI::parseReal,              NULL,
{ "RailgunExtraDistance",         INI::parseReal,              NULL,
{ "RailgunMaxDistance",           INI::parsePositiveNonZeroReal, NULL,
```

- Added RailgunRadius. Determines the Radius or Size of the Railgun for each Check. Defaults to: 0.
- Added RailgunRadiusCheckPerDistance. Configure the distance in units to trigger each Radius Check. Each time a Radius Check is triggered, it also triggers the RailgunFX and RailgunOCL for the Railgun, if configured. By default, the RadiusCheckPerDistance defaults to the ‘PartitionCellSize’ in GameData.ini. Defaults to: 0 or ‘PartitionCellSize’.

- Added RailgunInfatryRadius. Since Infantry often has a different height configuration when Compared to Tanks or Structures, the Checks can miss if configured with “RailgunIsLinear”. When declaring this variable, it will use a different checking radius for checking Infantry. Defaults to: 0 or None.
- Added RailgunExtraDistance. Configured in Distance Units, and enables Railgun to go beyond Target’s Position by the configured Distance. This sets the Targeted Position further and adjusted towards ‘RailgunPiercesBehind’ checking. Can also be configured with Negative Values to reduce the Railgun Distance. Defaults to: 0.
- Added RailgunMaxDistance. Configure the Maximum Distance the Railgun is able to reach. Defaults to: 0 or Unlimited.

```
{ "RailgunDamageType",                DamageTypeFlags::parseSingleBitFromINI, NULL,
{ "RailgunDeathType",                 INI::parseIndexList,
{ "RailgunOverrideDamageTypeFX",      DamageTypeFlags::parseSingleBitFromINI, NULL,
{ "RailgunCustomDamageType",          INI::parseAsciiString,          NULL,
{ "RailgunCustomDeathType",           INI::parseAsciiString,          NULL,
```

- Added RailgunDamageType. Determines the DamageType used by the Railgun if Configured. If not configured, use the Weapon’s DamageType. Defaults to: Not Configured.
- Added RailgunDeathType. Determines the DeathType used by the Railgun if Configured. If not configured, use the Weapon’s DeathType. Defaults to: Not Configured.
- Added RailgunOverrideDamageTypeFX. Determines the OverrideDamageTypeFX used by the Railgun if Configured. If not configured, use the Weapon’s OverrideDamageTypeFX. Defaults to: Not Configured.
- Added RailgunCustomDamageType. Determines the CustomDamageType used by the Railgun if Configured. If not configured, use the Weapon’s CustomDamageType. Defaults to: Not Configured.
- Added RailgunCustomDeathType. Determines the CustomDeathType used by the Railgun if Configured. If not configured, use the Weapon’s CustomDeathType. Defaults to: Not Configured.

```
{ "RailgunFX",                parseAllVetLevelsFXList,        NULL,
{ "RailgunOCL",               parseAllVetLevelsAsciiString,   NULL,
{ "RailgunVeterancyFX",       parsePerVetLevelFXList,         NULL,
{ "RailgunVeterancyOCL",      parsePerVetLevelAsciiString,    NULL,
```

- Added RailgunFX. Determines the FX used by the Railgun for each Distance Check if Configured. Defaults to: Not Configured.

- Added RailgunOCL. Determines the OCL to be Spawned by for each Distance Check if Configured. Defaults to: Not Configured.
- Added RailgunVeterancyFX. Determines the FX used by the Railgun for each Distance Check for different Veterancy Levels. Declared in the same format as VeterancyFireFX. Defaults to: Not Configured.
- Added RailgunVeterancyOCL. Determines the OCL to be Spawned by for each Distance Check if Configured. Declared in the same format as VeterancyFireOCL. Defaults to: Not Configured.

- SHIELDED_TARGET Custom Status Feature:

```
{ "ShielderProtectionType",                 INI::parseProtectionTypeFlags,  NULL,
{ "IgnoresShielder",                        INI::parseBool,         NULL,
```

- Added ShielderProtectionType. Works as a filter for the “SHIELDED_TARGET” feature. Declared “ALL” or “NONE” first with “+-”, “BULLETS”, “LASER”, and “PROJECTILES” later. This filters the type of Weapons the Shielder is able to redirect. Defaults to: ALL or +BULLETS +LASER +PROJECTILES.
- LASER consists of Weapons with a LaserName for the weapon.
- BULLETS consists of Weapons without a ProjectileObject and does not have a LaserName.
- PROJECTILES consists of Weapons that have a ProjectileObject.
- Does not redirect LeechRangeWeapons or Weapons that detonate at their positions.
- Added IgnoresShielder. If set to ‘Yes, that weapon does not consider the effects of “SHIELDED_TARGET” on the Unit. Defaults to: No.

- Subdual Damage Features:

```
{ "SubduedProjectileNoDamage",             INI::parseBool,         NULL,
{ "SubdualMissileAttractor",                INI::parseBool,         NULL,
```

- Added SubduedProjectileNoDamage. Subdued projectile still deals damage at its location after scattering, setting this to ‘Yes’ disables the projectile from dealing damage after Subdued, but doesn’t stop it from detonating. Defaults to: No.
- Added SubdualMissileAttractor. If Projectiles are Subdued with this mechanic, they are drawn from the original target towards the Unit instead of scattering from the intended position.

“SHIELDED_TARGET” feature stacks with SubdualMissileAttractor. However, SubdualMissileAttractor will redirect Shielded projectiles in range towards the Attractor.

The main difference between “SHIELDED_TARGET” feature and SubdualMissileAttractor is that “SHIELDED_TARGET” can redirect non-Projectile weapons while SubdualMissileAttractor can only affect Projectiles in range that are Subduable, but can filter the KindOf and Armors for the Projectiles it affects.

```
{ "WeaponIsSubdual",                    parseAllVetLevelsBool,          NULL,
{ "VeterancyWeaponIsSubdual",           parsePerVetLevelBool,           NULL,
{ "SubdualDealsNormalDamage",           parseAllVetLevelsBool,          NULL,
{ "VeterancySubdualDealsNormalDamage",  parsePerVetLevelBool,           NULL,
{ "SubdualDamageMultiplier",            parseAllVetLevelsReal,          NULL,
{ "VeterancySubdualDamageMultiplier",   parsePerVetLevelReal,           NULL,
{ "SubdualForbiddenKindOf",             KindOfMaskType::parseFromINI,   NULL,
```

- Added WeaponIsSubdual which overrides the Weapon to deal Subdual Damage with its DamageType for damage calculation rather than requiring to configure SUBDUAL DamageTypes. If this is set to ‘Yes’, the weapon will not deal damage to a Unit's Health but will deal Subdual damage like Subdual DamageTypes.
- Added VeterancyWeaponIsSubdual, which allows the customization of WeaponIsSubdual for Veterancy Levels.
- Added SubdualDealsNormalDamage which when set to ‘Yes’, considers the weapon to deal damage to the health in addition to the Subdual Damage. This is configured only when WeaponIsSubdual is set to ‘Yes’ and is disabled for weapons with SUBDUAL DamageTypes.
- Added VeterancySubdualDealsNormalDamage, which allows the customization of SubdualDealsNormalDamage for Veterancy Levels.
- Added SubdualDamageMultiplier which multiplies the Subdual damage it deals compared to its default damage. It uses decimal values like 3.3 instead of percentages. This also works for weapons with SUBDUAL DamageTypes.
- Added VeterancySubdualDamageMultiplier, which allows the customization of SubdualDamageMultiplier for Veterancy Levels.
- Added SubdualForbiddenKindOf which disables the weapon from dealing Subdual Damage towards any certain KindOfs; or for weapons with SUBDUAL DamageTypes, disables the Weapon from using towards the declared KindOfs.

- NEW Custom Subdual Damage Types:

```
{ "SubdualCustomType",                    INI::parseAsciiString,  NULL,
```

- Added SubdualCustomType which enables the Weapon to deal a Customizable Type of Subdual Damage separated from Subdual types. Configuring this with any word/string will register the weapon to have a Separated Subdual Mechanic, overriding the typical Subdual mechanic of SUBDUAL DamageTypes or WeaponIsSubdual.
- Custom Subdual Types have more configurations and features than Subdual Mechanic. An Object can receive multiple kinds of SubdualCustomTypes, but a Weapon can only have one kind of SubdualCustomType. If both weapons have the same SubdualCustomType, they will add Damage to the on top of each other, the same way as how Subdual mechanics work.
- Custom Subdual Types are affected by 'SubdualDealsNormalDamage', 'SubdualDamageMultiplier', 'SubdualForbiddenKindOf', 'SubduedProjectileNoDamage', and 'SubdualMissileAttractor' features mentioned above.
- They also consider CustomSubdualDamageCap, CustomSubdualDamageHealRate, and CustomSubdualDamageHealAmount for Subdual damage calculation if they are configured. (See Active Body below)

```
{ "CustomSubdualTintStatus",              parseAllVetLevelsTintStatus,        NULL,
{ "CustomSubdualCustomTintStatus",        parseAllVetLevelsAsciiString,       NULL,
{ "CustomSubdualHasDisable",              parseAllVetLevelsBool,          NULL,
{ "CustomSubdualHasDisableProjectile",    parseAllVetLevelsBool,          NULL,
{ "CustomSubdualClearOnTrigger",          parseAllVetLevelsBool,          NULL,
{ "CustomSubdualDoStatus",                parseAllVetLevelsBool,          NULL,
{ "CustomSubdualOCL",                     parseAllVetLevelsAsciiString,   NULL,
```

- Added CustomSubdualTintStatus. This enables the Tint of the Object to change when being Subdued. Custom Subdual Types do not use WeaponStatusTintStatus for painting Tint during Subduing. The Original Subdual mechanic paints the Object with TINT_STATUS_GAINING_SUBDUAL_DAMAGE. This enables it to be overridden. Defaults to: TINT_STATUS_GAINING_SUBDUAL_DAMAGE.
- Added CustomSubdualCustomTintStatus. Enables the SubdualCustomType to use CustomTintStatus. Overrides CustomSubdualTintStatus when configured.
- Added CustomSubdualHasDisable, which when set to 'No', makes the Custom Subdual Type to not Disable the affected Objects after triggering the Subdual. Defaults to: Yes.

- Added CustomSubdualHasDisableProjectiles, which when set to ‘No’, makes the Custom Subdual Type to not Jam or Attract the affected Projectiles after triggering the Subdual. Defaults to: Yes.
- Added CustomSubdualClearOnTrigger, which resets the Subdual Damage of the Custom Subdual Type to near ‘0’ after triggering the Subdual. This is better suited for extended features below. Defaults to: No.
- Added CustomSubdualDoStatus, which allows the Custom Subdual Type to apply DamageStatus or CustomDamageStatus (see feature above) every time while dealing Subdual Damage after the Unit is Subdued. Defaults to: No.
- Added CustomSubdualOCL, which allows the Custom Subdual Type to create OCL every time while dealing Subdual Damage after the Unit is Subdued. Defaults to: None.

```
{ "VeterancyCustomSubdualTintStatus",           parsePerVetLevelTintStatus,         NULL,
{ "VeterancyCustomSubdualCustomTintStatus",     parsePerVetLevelAsciiString,        NULL,
{ "VeterancyCustomSubdualHasDisable",           parsePerVetLevelBool,           NULL,
{ "VeterancyCustomSubdualHasDisableProjectile", parsePerVetLevelBool,           NULL,
{ "VeterancyCustomSubdualClearOnTrigger",       parsePerVetLevelBool,           NULL,
{ "VeterancyCustomSubdualDoStatus",             parsePerVetLevelBool,           NULL,
{ "VeterancyCustomSubdualOCL",                  parsePerVetLevelAsciiString,    NULL,
```

- Added VeterancyCustomSubdualTintStatus, which allows the customization of CustomSubdualTintStatus for Veterancy Levels.
- Added VeterancyCustomSubdualCustomTintStatus, which allows the customization of CustomSubdualCustomTintStatus for Veterancy Levels.
- Added VeterancyCustomSubdualHasDisable, which allows the customization of CustomSubdualHasDisable for Veterancy Levels.
- Added VeterancyCustomSubdualHasDisableProjectiles, which allows the customization of CustomSubdualHasDisableProjectiles for Veterancy Levels.
- Added VeterancyCustomSubdualClearOnTrigger, which allows the customization of CustomSubdualClearOnTrigger for Veterancy Levels.
- Added VeterancyCustomSubdualDoStatus, which allows the customization of CustomSubdualDoStatus for Veterancy Levels.
- Added VeterancyCustomSubdualOCL, which allows the customization of CustomSubdualOCL for Veterancy Levels.

```
{ "CustomSubdualDisableType",                DisabledMaskType::parseSingleBitFromINI, NULL,
{ "CustomSubdualDisableTint",                TintStatusFlags::parseSingleBitFromINI,     NULL,
{ "CustomSubdualDisableCustomTint",          INI::parseAsciiString,          NULL,
{ "CustomSubdualRemoveSubdualTintOnDisable", INI::parseBool,                 NULL,
{ "CustomSubdualDisableSound",               INI::parseAsciiString,          NULL,
{ "CustomSubdualDisableRemoveSound",         INI::parseAsciiString,          NULL,
```

- Added CustomSubdualDisableType, which allows the Disable Type for Custom Subdual Type to be customizable. Defaults to: DISABLED_SUBDUED.
- Added CustomSubdualDisableTint. You are able to customize the Tint for the Disable Type. Defaults to: TINT_STATUS_INVALID. However, when configured with Disabled types that uses TINT_STATUS_DISABLED while it is configured TINT_STATUS_INVALID, the system will use TINT_STATUS_DISABLED.
- Added CustomSubdualDisableCustomTint. Enables the SubdualCustomType's Disable mechanic to use CustomTintStatus. Overrides CustomSubdualDisableTint when configured.
- Added CustomSubdualRemoveSubdualTintOnDisable. When enabled,the SubdualCustomType does not paint (while Removing) its Subdual Tint on the Object while it is Disabled. This is VERY USEFUL to mitigate Tint Color Correction Bug since assigning two Tints at the same time will cause the Drawable to Bug out. Defaults to: No.
- Added CustomSubdualDisableSound. If configured, plays a sound after a Unit has been Disabled with the Weapon's Custom Subdual Type. Defaults to: None.
- Added CustomSubdualDisableRemoveSound. If configured, plays a sound after a Unit has been Cleared from begin Disabled with the Weapon's Custom Subdual Type. Defaults to: None.
- Bug: Tint Status may overlap and permanently change the coloration of the affected unit. To mitigate this issue, use CustomSubdualRemoveSubdualTintOnDisable.

- NEW Shrapnel Weapons:

```
{ "ShrapnelCount",              INI::parseInt,
{ "ShrapnelWeapon",         INI::parseWeaponTemplate,                                NULL
{ "ShrapnelDoesNotRequireVictim",               INI::parseBool,                NULL,
{ "ShrapnelCanTargetStealth",               INI::parseBool,                  NULL,
{ "ShrapnelTargetMask",         INI::parseBitString32,  TheWeaponAffectsMaskNames,
```

- Added ShrapnelWeapon. Inspired by one of the famous Mechanics of RA2, Shrapnel Weapons is an implementation attempt into General's Engine.

Works the same way as RA2, and requires the Weapon to hit a Target to generate Shrapnels. Shrapnels are then retargeted to Neutral or Enemy Objects within the Template's WeaponRange. Extra Shrapnels are targeted towards a random direction with the Attack Range as the distance.

NOTE: Like 'HistoricBonusWeapon', the Templates used are required to be configured Above the Weapon using it. For ShrapnelWeapons using a ProjectileObject with MissileAIUpdate, they are required to be configured with a Locomotor that has a High TurnRate to be able to target accurately.
Defaults to: Not configured.

- Added ShrapnelCount. Configure the amount of Shrapnels able to generate. Defaults to: 0.
- Added ShrapnelDoesNotRequireVictims. Configure whether the Weapon is able to generate Shrapnels without hitting an Object. Defaults to: No.
- Added ShrapnelCanTargetStealth. Configure the Shrapnels generated from the Weapon can consider a Stealthed or Disguised Object as a Target. Defaults to: No.
- Added ShrapnelTargetMask. Declared in the same format as RadiusDamageAffects. Configure the relationship between the User and the Object the Shrapnel is able to Target. Defaults to: NEUTRAL, and ENEMIES.

- NEW ROF Moving Penalty:

```
{ "ROFPenaltyWhenMoving",                INI::parsePercentToReal,          NULL
{ "ROFMovingScalesWithMovingSpeed",      INI::parseBool,         NULL,
{ "ROFMovingScalingSpeedMax",            INI::parseVelocityReal,     NULL,
```

- Added ROFPenaltyWhenMoving. Rate of Fire can be configured to receive a Penalty. If configured, the penalty is calculated when the Object is Firing while Moving. Configured in (%) and acts as a PlusMinus towards the Multiplier to the current Delay Duration for next time when Fire. Can be configured with Negative Values for faster ROF. Defaults to: 0%, or not configured.
- Added ROFMovingScalesWithMovingSpeed. Configure 'ROFPenaltyWhenMoving' to Scale with the Object's Current Moving Speed. If Set to 'Yes', the value of 'ROFPenaltyWhenMoving' is diminished according to the Current Movespeed, and providing the full value when reaching Max Speed. Defaults to: No.

- Added ROFMovingScalingSpeedMax. Adjust the MaxSpeed Limit for calculating the Value towards ‘ROFMovingScalesWithMovingSpeed’. The maximum ‘ROFMovingScalesWithMovingSpeed’ value cannot exceed what is declared in ‘ROFPenaltyWhenMoving’. Defaults to: No.

- NEW Line of Sight Bypass:

```
{ "UserBypassLineOfSight",                INI::parseBool,                NULL,
{ "UserIgnoresObstacles",                 INI::parseBool,                NULL,
```

- Added UserBypassLineOfSight. Allows the User of this Weapon to BypassLineOfSight for both Movement and Weapon Targeting. Defaults to: No.
- Added UserIgnoresObstacles. Allows the User of this Weapon to Ignore Stationary or Blocking Targets for both Movement and Weapon Targeting. Defaults to: No.

- CrateCollide Compatibility Features:

```
{ "RejectKeys",                           INI::parseAsciiStringVector,
```

```
{ "ClearsParasite",                       INI::parseBool,          NULL,
{ "ClearsParasiteKeys",                   INI::parseAsciiStringVector,
```

- Added RejectKeys. If configured with any reject keys, the weapon cannot be used against Targets with the Reject Keys granted from Crate Collide Modules. Can be configured with more than one Reject Keys. Defaults to: Not configured.
- Added ClearsParasite. Hitting an Object with this weapon clears Parasite from the Object. Clearing Parasites remove them as if ejecting them, or destroying them if “DestroyOnClear” is set to ‘Yes’. Does nothing if the Object is not Equipped with an Object that has EquipCrateCollide module with IsParasite set to “Yes”. Defaults to: No.
- Added ClearsParasiteKeys. If a Parasite has a ParasiteKey, the weapon requires to have ‘ClearParasite’ set to ‘Yes’, along with the matching Key in the ‘ClearsParasiteKey’ while hitting an Object to clear Parasite from the Object. Can be configured with multiple keys, and they are kept as a List. Defaults to: Not configured.

- UNRESISTABLE Death Prevention Bypass Features:

```
{ "WeaponNotAbsoluteKill",                INI::parseBool,          NULL,
```

- Added WeaponNotAbsoluteKill. When paired with ‘UNRESISTABLE’ DamageType, the damage dealt does not disable the target’s kill prevention effects for triggering. Does nothing outside of ‘UNRESISTABLE’ DamageType. Defaults to: No.

  Currently, there are only two features that use this mechanic: “UndeadBody” and “HighlanderBody” both of which have death prevention effects on Trigger.

Armors:

- Armor Weapon Bonus addition:

```
enum Field
{
    DAMAGE = 0,
    RADIUS,
    RANGE,
    RATE_OF_FIRE,
    PRE_ATTACK,
    ARMOR,

    FIELD_COUNT // keep last
};
```

- WeaponBonus and CustomWeaponBonus now considers ARMOR bonuses for WeaponBonus types.
- They can be either declared globally or within a Weapon. However, both conditions require a Weapon to trigger.
- While using a weapon with ARMOR bonus type, the unit will receive damage reduction (less than 100%,) or vulnerability (more than 100%) when under the declared Weapon Condition.
- This is only limited to units with weapons, (even a dummy weapon would do.) For implementing this feature without requiring Weapons, see below.

- Armor Template:

```
{ "Armor", ArmorTemplate::parseArmorCoefficients, NULL, 0 },
{ "ArmorMult", ArmorTemplate::parseArmorMultiplier, NULL, 0 },
{ "ArmorBonus", ArmorTemplate::parseArmorBonus, NULL, 0 },
```

- Added ArmorBonuses, which function like WeaponBonus, but can register either StatusTypes, WeaponBonusConditions or CustomStatusTypes/CustomWeaponBonusConditions (See Below).
- They can be stacked altogether multiple times.
- Declare them with either a Status Type, Weapon Bonus Condition Type or Custom Status Type first, and then a bonus in percentage(%) second.
- For the Custom Status / Custom Weapon Bonus Check, the system will check first if it's within the Unit's Custom Weapon Bonus Conditions List. If it's within the Unit's Custom Weapon Bonus Condition, or it has been applied before, it will treat it as a Custom Weapon Bonus Type.

- Only if it's not within the List of Custom Weapon Bonus Conditions, then it will proceed to check it within the Unit's Custom Status Types List.
- It will check for all the satisfied conditions declared within the bonuses.

For Example:

Armor BattleBusTruckArmor

ArmorBonus = DEMORALIZED 1000%

ArmorBonus = CustomBonus1 25%

ArmorBonus = CustomStatus1 50%

ArmorBonus = CustomStatus2 150%

ArmorBonus = BATTLEPLAN_HOLDTHELINE 0%

ArmorBonus = BATTLEPLAN_SEARCHANDDESTROY 10%

ArmorBonus = FRENZY3 10%

End

- The system will check for every Status Types, Weapon Bonus Condition Types and Custom Status Type existing on the Armor Template, then scale the damage according to each condition type.
- For example, if the Object is affected by both Search and Destroy and Frenzy 3, it will check for both SEARCHANDDESTROY and FRENZY3 for scaling damage: 1.0 * 0.1 * 0.1 = 0.01 = 1%.
- if the Object is affected by all of the statuses above, then the result will be 0%, since BATTLEPLAN_HOLDTHELINE is declared at 0%.

Units/Objects:

- More Generic KindOfs.
- These KindOfs do nothing within the Source Code and can be defined to any objects. These are written to define more versatile variables to be used by KindOfs features, as there is no custom KindOf defining feature yet.

// UNIVERSAL TYPE

GARRISONABLE, GARRISON_SPECIFIC, SUBTERRANEAN, ANTI_INFANTRY, ANTI_TANK,ANTI_NAVAL, ANTI_STRUCTURE, ANTI_GARRISON, ANTI_MINES, FLAMER, POISONER, EMP_USER, DISABLER, SUBDUAL_USER, SUBDUAL_INF, SUBDUAL_VEH, SUBDUAL_BUILDING, SIGNAL_JAMMER, DEFECTOR, STEALER, CAPTURER, CONTROLLER, LINK_MASTER, LINK_SLAVE, BUILDER, RADAR_GIVER, CANT_ATTACK, SLOW

// Different levels of Stealth

STEALTHED2, STEALTHED3, STEALTHED4, STEALTHED5, STEALTHED6, STEALTHED7, STEALTHED8

// Different levels, one of the examples were Upgrades and Salvaged Crates. They could also be used for different Tiered Units.

LEVEL2, LEVEL3, LEVEL4, LEVEL5, LEVEL6, LEVEL7, LEVEL8

// INFANTRIES

PILOT, ENGINEER, HIJACKER, SABOTEUR, SPY, SNIPER, ELITE_SNIPER, SPECIAL_FORCES, ROBOT_INFANTRY, HEAVY_INFANTRY, SUPER_INFANTRY, RAMBO

// VEHICLES

LIGHT_VEHICLE, BIKE, CAR, ROBOT, MECH, TITAN, POWERED_TANK, MINE_LAYER, DRONE_LAUNCHER, MISSILE_LAUNCHER, UNIQUE_VEHICLE, FOUR_WHEELED, LORRY

// AIRCRAFT

STRIKER, BOMBER, STEALTH_AIRCRAFT, SUPPORT_AIRCRAFT, AIR_TRANSPORT, APACHE, HEAVY_APACHE, HELI_CARRIER

// BUILDINGS

CAN_CHARGE (Defining those like Prism Tower and Tesla Coil), DEFENSE_STRUCTURE, TECH_STRUCTURE, MEGA_STRUCTURE,

MEGA_BASE_DEFENSE, FS_MEGA_POWER, EMPULSE_CANNON, TUNNEL_NETWORK, DERELICT

// PROJECTILES

SUPERSONIC, HYPERSONIC, LARGE_PROJECTILE, BULLET, SPRAY, TANK_SHELL, ENERGY_BLAST, AIR_MISSILE, PARABOMB, ARTILLERY_SHELL, BIG_MISSILE, AIRTOAIR, AIRTOGROUND, GROUNDTOAIR, GROUNDTOGROUND, GUIDED, PATHED, DETONATE, AROUND_SELF, AOE, AOE_SPREAD, SHRAPNEL, FRAGMENT, SCATTER_MISL, MULTI_TARGET, REQUIRES_TARGET, REQUIRES_TARGET2, REQUIRES_TARGET3, REQUIRES_TARGET4, REQUIRES_TARGET5, REQUIRES_TARGET6, REQUIRES_TARGET7, REQUIRES_TARGET8

// WARHEADS

NUCLEAR, NEUTRON_BLAST, EXPLOSIVE, HIGH_EXPLOSIVE, PLASMA_WARHEAD, DELAYED, FUELAIR_BOMB, SPAWNER, ASSISTER, STATUS_GIVER, NON_DAMAGE, ANTI_MATTER, VACUUM_WARHEAD, IS_EARTHQUAKE, NON_PHYSICAL, FREEZER, ORBITAL_LASER

// NON-SELECTABLES

MINE_SPAWNER, MONEY_CRATE, SALVAGE_CRATE, SUPER_SALVAGE_CRATE, STATUS_CRATE, BUFF_CRATE, DEBUFF_CRATE, SUPER_CRATE, RARE_CRATE, EPIC_CRATE, CRATE_GIVER, UPGRADE_GIVER, UPGRADE_REMOVER, NON_INTERACTABLE (Such as Dummy), NON_SELECTABLE, OUT_OF_BOUNDS, INVISIBLE, NO_MODEL

// ENVIRONMENTAL (For those insane enough to use KindOfs to design immersive environments)

GRASS_ENV, WATER_ENV, WOOD_ENV, WAVE_ENV, STONE_ENV, WIND_ENV, PHYSICS_ENV, ORGANIC, ANIMAL, MAMMAL, FISH, STEEL, LIQUID, REACTIVE, FURNITURE, BENCH, BRICKS, SIGNS, FLAT_DECOR, STONEWALL, HOUSE, SKYSCRAPER, TRAIN, DECOR

- Negative Prerequisites:

```
// Prerequisite to Deny the Unit from Building
{ "NegativePrerequisites",            ThingTemplate::parseNegativePrerequisites,  0, 0 },
{ "HideNegativePrerequisites",        INI::parseBool,     NULL, offsetof( ThingTemplate, m_negprereqHideInfo ) },
```

> Added NegativePrequisites, which functions similarly to Prerequisites but disables the Unit from Production if Built or Present.

> Negative Prerequisites Tooltips always show up, whether the Prerequisites are present or not.

> To prevent the Tooltip of Negative Prerequisites from showing up, set HideNegativePrequisites = Yes for the unit.

```
CONTROLBAR:NegativeRequirements
"Negative Requirements: %s"
END

CONTROLBAR:GeneralsPromotionNeg
"Promotion Negative: %s"
END
```

> Here are the Tooltips required for the Negative Prerequisites to properly show.

- MaxSimultaneous Properties Expansion:

```
{ "MaxSimultaneousLinkObjects",         INI::parseAsciiStringVector,              NULL,       o
{ "MaxSimultaneousOfTypeDifficulty",        ThingTemplate::parseMaxSimultaneousOfTypeDifficulty,
{ "MaxSimultaneousOfTypeDifficultyAIOverride",      ThingTemplate::parseMaxSimultaneousOfTypeDiff
{ "MaxSimultaneousOfTypeCustomMessage",         INI::parseAndTranslateLabel,              NULL,
```

- Added MaxSimultaneousLinkObjects:

- A more robust version of MaxSimultaneousLinkKey.
- Enables other units to consider as another unit towards MaxSimultaneousOfType count while considering Different Skins/BuildVariations (implemented with IsEquivalentTo).
- Unlike MaxSimultaneousLinkKey, this feature also counts the Objects that are currently in Production Queues.
- It also only considers Other Objects when adding towards the Unit's Own MaxSimultaneousOfType count.
- The other Unit within MaxSimultaneousLinkObjects is Not Considered as a Count towards its own MaxSimultaneousOfType.
- This can be individually implemented to work like MaxSimultaneousLinkKey.

- Added MaxSimultaneousOfTypeDifficulty:

- You can customize the MaxSimultaneousOfType for different levels of difficulty. Namely: EASY, NORMAL, and HARD. They register in the format below.

Example,

MaxSimultaneousOfTypeDifficulty = EASY 2 HARD 1

- If you're in Campaign, and the Campaign is on Hard Difficulty, you will only be able to build 1 unit, whereas 2 in Easy. Uses Default MaxSimultaneousOfType values at Normal (If MaxSimultaneousOfType is not configured, it will be unlimited.)

- Does nothing if the Player plays Skirmish.

- MaxSimultaneousOfTypeDifficulty affects AI difficulty in both Campaign and Skirmish. To override the list for AI exclusively, use MaxSimultaneousOfTypeDifficultyAIOverride. The way it's declared is like MaxSimultaneousOfTypeDifficulty.

Example,

MaxSimultaneousOfTypeDifficultyAIOverride = HARD 10

The declared value enables AI to build up to 10 units at Hard difficulty and default values at either Easy or Normal. To customize the values for Easy, you will need to declare MaxSimultaneousOfTypeDifficultyAIOverride = EASY 2 HARD 10 to override the default value.

- Added MaxSimultaneousOfTypeCustomMessage:

- You can now override the MaxSimultaneousOfType Tooltip on the Control Bar when the MaxSimultaneousOfType for a unit is reached.

Example, declaring:

Object

MaxSimultaneousOfTypeCustomMessage = GUI:ParkingPlacesFull

End

- It will show the message stated in the String Table instead.

- Underpower/Power Outrage features:

- NOTE: When there is a Power Outrage (Consuming Energy being more than Supply Energy), only POWERED Objects are affected by the Power Outrage. Objects that have POWERED_TANK only within their KindOfs (Newly added above,) are NOT affected, but they can be manually Triggered to be Underpowered with DISABLE_POWER Command.

```
{ "DisabledWhenUnderpowered",          INI::parseBool,     NULL, offsetof( ThingTemplate, m_setDi
{ "DisabledTypeWhenUnderpowered",      DisabledMaskType::parseSingleBitFromINI, NULL, offsetof( T
{ "StatusUnderPowered",         ObjectStatusMaskType::parseFromINI, NULL, offsetof( ThingTemplate,
{ "CustomStatusUnderPowered",   INI::parseAsciiStringVector, NULL, offsetof( ThingTemplate, m_cust
{ "WeaponBonusUnderPowered",    INI::parseWeaponBonusVector, NULL, offsetof( ThingTemplate, m_bonu
{ "CustomWeaponBonusUnderPowered",          INI::parseAsciiStringVector, NULL, offsetof( ThingTemp
{ "TintStatusUnderPowered",         TintStatusFlags::parseSingleBitFromINI,     NULL, offsetof( Th
{ "CustomTintStatusUnderPowered",   INI::parseAsciiString,  NULL, offsetof( ThingTemplate, m_custo
{ "ModelConditionUnderPowered", ModelConditionFlags::parseFromINI, NULL, offsetof( ThingTemplate,
```

- Added DisabledWhenUnderpowered. Configure whether the Object is to be Disabled with a DisabledType when Underpowered. Defaults to: Yes.
- Added DisabledTypeWhenUnderpowered. Configures the DisabledType for the Underpowered Object while Disabled. Defaults to: DISABLED_UNDERPOWERED.
- Added StatusUnderpowered. Sets the Statuses for the Object with the declared variables when it is Underpowered. Clear the Statuses when it is Powered. Defaults to: None.
- Added CustomStatusUnderpowered. Sets the Custom Statuses for the Object with the declared variables when it is Underpowered. Clear the Custom Statuses when it is Powered. Defaults to: None.
- Added WeaponBonusUnderpowered. Sets the Weapon Bonuses for the Object with the declared variables when it is Underpowered. Clear the Weapon Bonuses when it is Powered. Defaults to: None.
- Added CustomWeaponBonusUnderpowered. Sets the Custom Weapon Bonuses for the Object with the declared variables when it is Underpowered. Clear the Custom Weapon Bonuses when it is Powered. Defaults to: None.
- Added TintStatusUnderpowered. Sets the Custom Tint Status for the Object with the declared variables when it is Underpowered. Clear the Custom Tint Status when it is Powered. Defaults to: None.
- Added CustomTintStatusUnderpowered. Sets the Custom Tint Status for the Object with the declared variables when it is Underpowered. Clear the Custom Tint Status when it is Powered. Defaults to: None.
- Added ModelConditionUnderpowered. Sets the Model Condition for the Object with the declared variables when it is Underpowered. Clear the Model Condition when it is Powered. Defaults to: None.

- WeaponSets

```
{ "Weapon", WeaponTemplateSet::parseWeapon, NULL, 0 },
{ "AutoChooseSources",  WeaponTemplateSet::parseAutoChoose, NULL, 0 },
{ "PreferredAgainst", WeaponTemplateSet::parsePreferredAgainst, NULL, 0 },
{ "ShareWeaponReloadTime", INI::parseBool, NULL, offsetof( WeaponTemplateSet, m_isReloadTimeShared ) },
{ "WeaponLockSharedAcrossSets", INI::parseBool, NULL, offsetof( WeaponTemplateSet, m_isWeaponLockSharedA
{ "WeaponReloadSharedAcrossSets", INI::parseBool, NULL, offsetof(WeaponTemplateSet, m_isWeaponReloadShar
{ "WeaponChoiceCriteria", INI::parseIndexList, TheWeaponChoiceCriteriaNames, offsetof( WeaponTemplateSet
```

- Restored WeaponChoiceCriteria to be able to find Longest Range when choosing a weapon to use instead of using the weapon with the Highest Damage Amount.
- Currently has two options: ‘DAMAGE’, and ‘RANGE’.
- DAMAGE is the default criteria which uses the highest Damage Amount for all WeaponSets, while RANGE determines the weapon with the Longest Range to use.
- Note: If a Weapon is still not ready, (such as, Reloading or Out of Clip,) the System will find another Weapon to use for the time being. To prevent the WeaponSet from switching, see PriorityWithinAttackRange and PriorityOutsideAttackRange in the Weapons Section.

- WeaponBonusUpdate:

```
{ "RequiredAffectKindOf",       KindOfMaskType::parseFromINI,       NULL, offs
{ "ForbiddenAffectKindOf",  KindOfMaskType::parseFromINI,       NULL, offsetof
{ "AffectsTargets", INI::parseBitString32,  TheWeaponAffectsMaskNames, offseto
{ "AffectAirborne", INI::parseBool, NULL, offsetof(WeaponBonusUpdateModuleData
{ "BonusDuration",                  INI::parseDurationUnsignedInt,  NULL, offs
{ "BonusDelay",                         INI::parseDurationUnsignedInt,  NULL,
{ "BonusRange",                         INI::parseReal,
{ "BonusConditionType",         INI::parseIndexList,    TheWeaponBonusNames, o
{ "CustomBonusConditionType",       INI::parseAsciiString,  NULL, offsetof( We
{ "TintStatusType",         TintStatusFlags::parseSingleBitFromINI, NULL, offs
{ "CustomTintStatusType",       INI::parseAsciiString,  NULL, offsetof( Weapon
```

- Added Support for CustomBonusConditionType, and CustomTintStatusType.
- CustomBonusConditionType overrides the BonusConditionType. If declared, it will grant the CustomBonusConditionType instead of the BonusConditionType.
- CustomTintStatusType overrides the TintStatusType. If declared, it will use the CustomTintStatusType instead of the TintStatusType.

- SpecialAbilityUpdate:

```
{ "DeleteUserOnExecute",                INI::parseBool,
{ "FXOnExecute",                        INI::parseFXList,                   NULL
{ "OCLOnExecute",                       INI::parseObjectCreationList,       NULL
```

- Added DeleteUserOnExecute. If set to ‘Yes’, it will delete the object after triggering its SpecialAbilityUpdate abilities.
- With proper implementation, you can bring in a lot of other unit features from the CNC series.
- It uses the same form of Deletion used by DeletionUpdate, which would not inform or alert the system on delete.
- Added FXOnExecute and OCLOnExecute. If set, spawns the respective FX and OCL on the Object after triggering the SpecialAbilityUpdate ability.

```
{ "KindOf",                 KindOfMaskType::parseFromINI,
{ "ForbiddenKindOf",        KindOfMaskType::parseFromINI,
```

- Added KindOf and ForbiddenKindOf. These function currently works for SpecialPowerTemplate that has a SpecialPower of SPECIAL_DISGUISE_AS_VEHICLE, SPECIAL_DEFECTOR, or SPECIAL_MISSILE_DEFENDER_LASER_GUIDED_MISSILES.
- Overrides the KindOf for targeting with the listed SpecialPowers, allowing it to target not just VEHICLES or limit its target KindOfs. Defaults to: None. When None is configured, the KindOf and ForbiddenKindOf are configured below.

  KindOf:
  - SPECIAL_DISGUISE_AS_VEHICLE: ‘VEHICLE’
  - SPECIAL_MISSILE_DEFENDER_LASER_GUIDED_MISSILES: ‘VEHICLE’
  - SPECIAL_DEFECTOR: ALL

  ForbiddenKindOf:
  - SPECIAL_DISGUISE_AS_VEHICLE: ‘AIRCRAFT’, ‘BOAT’, and ‘CLIFF_JUMPER’
  - SPECIAL_MISSILE_DEFENDER_LASER_GUIDED_MISSILES: NONE
  - SPECIAL_DEFECTOR: ‘STRUCTURE’

  NOTE: SPECIAL_DISGUISE_AS_VEHICLE cannot target Objects with ‘RailroadBehavior’.

```
{ "WeaponSlot",                 INI::parseQuotedAsciiString,        NULL, offsetof
{ "AffectsTargets",             INI::parseBitString32,  TheWeaponAffectsMaskNames,
```

- Added WeaponSlot and AffectsTargets. These functions currently works for SpecialPowerTemplate that has a SpecialPower of SPECIAL_MISSILE_DEFENDER_LASER_GUIDED_MISSILES. The Special Power has been de-hardcoded to allow the modification of the usage of WeaponSlot and AffectsTargets to use.
- WeaponSlot is implemented to allow the customization of weapon usage for SPECIAL_MISSILE_DEFENDER_LASER_GUIDED_MISSILES. Defaults to: SECONDARY.
- AffectsTargets is implemented to allow SPECIAL_MISSILE_DEFENDER_LASER_GUIDED_MISSILES to target not just ‘ENEMIES’, but also ‘ALLIES’ and ‘NEUTRALS’. Defaults to: ENEMIES.

- Object Cursor Names:

```
{ "SelectingCursorName",              INI::parseAsciiString,
{ "MoveCursorName",                   INI::parseAsciiString,
{ "AttackMoveCursorName",             INI::parseAsciiString,
{ "WaypointCursorName",               INI::parseAsciiString,
{ "GenericInvalidCursorName",         INI::parseAsciiString,
{ "EnterCursorName",                  INI::parseAsciiString,
{ "EnterAggressiveCursorName",        INI::parseAsciiString,
{ "AttackCursorName",                 INI::parseAsciiString,
{ "ForceAttackCursorName",            INI::parseAsciiString,
{ "ForceAttackGroundCursorName",      INI::parseAsciiString,
{ "OutrangeCursorName",               INI::parseAsciiString,
{ "GetRepairAtCursorName",            INI::parseAsciiString,
{ "DockCursorName",                   INI::parseAsciiString,
{ "GetHealedCursorName",              INI::parseAsciiString,
{ "DoRepairCursorName",               INI::parseAsciiString,
{ "ResumeConstructionCursorName",     INI::parseAsciiString,
{ "SetRallyPointCursorName",          INI::parseAsciiString,
{ "SalvageCursorName",                INI::parseAsciiString,
{ "BuildCursorName",                  INI::parseAsciiString,
{ "InvalidBuildCursorName",           INI::parseAsciiString,
```

- Added "SelectingCursorName". Changes Selecting Cursor for the Object if configured. Defaults to: Not Configured.
- Added "MoveCursorName". Changes Move Cursor for the Object if configured. Defaults to: Not Configured.
- Added "AttackMoveCursorName". Changes Attack Move Cursor for the Object if configured. Defaults to: Not Configured.
- Added "WaypointCursorName". Changes Waypoint Cursor for the Object if configured. Defaults to: Not Configured.

- Added "GenericInvalidCursorName". Changes GenericInvalid Cursor for the Object if configured. Defaults to: Not Configured.
- Added "EnterCursorName". Changes Enter Cursor for the Object if configured. Defaults to: Not Configured.
- Added "EnterAggressiveCursorName". Changes EnterAggressive Cursor for the Object if configured. Defaults to: Not Configured.
- Added "AttackCursorName". Changes Attack Cursor for the Object if configured. Defaults to: Not Configured.
- Added "ForceAttackCursorName". Changes Force Attack Cursor for the Object if configured. Defaults to: Not Configured.
- Added "ForceAttackGroundCursorName". Changes Force Attack Ground Cursor for the Object if configured. Defaults to: Not Configured.
- Added "OutrangeCursorName". Changes Outrange Cursor for the Object if configured. Defaults to: Not Configured.
- Added "GetRepairAtCursorName". Changes Get Repair At Cursor for the Object if configured. Defaults to: Not Configured.
- Added "DockCursorName". Changes Dock Cursor for the Object if configured. Defaults to: Not Configured.
- Added "GetHealedCursorName". Changes Get Healed Cursor for the Object if configured. Defaults to: Not Configured.
- Added "DoRepairCursorName". Changes Do Repair Cursor for the Object if configured. Defaults to: Not Configured.
- Added "ResumeConstructionCursorName". Changes Resume Construction Cursor for the Object if configured. Defaults to: Not Configured.
- Added "SetRallyPointCursorName". Changes Set Rally Point Cursor for the Object if configured. Defaults to: Not Configured.
- Added "SalvageCursorName". Changes Salvage Cursor for the Object if configured. Uses Move Cursor if not configured. Defaults to: Not Configured.
- Added "BuildCursorName". Changes Build Cursor for the Object if configured. Defaults to: Not Configured.
- Added "InvalidBuildCursorName". Changes Invalid Build Cursor for the Object if configured. Defaults to: Not Configured.

```
{ "UseMyDockCursor",                    INI::parseBool,
{ "UseMyGetHealedCursor",               INI::parseBool,
{ "UseMyGetRepairAtCursor",             INI::parseBool,
{ "UseMyEnterCursor",                   INI::parseBool,
{ "UseMySalvageCursor",                 INI::parseBool,
```

- Added "UseMyDockCursor". Uses my Dock Cursor for the Dock Cursor instead of the Object's Dock Cursor when targeting me if I have "DockCursorName" configured. Defaults to: No.
- Added "UseMyGetHealedCursor". Uses my Get Healed Cursor for the Get Healed Cursor instead of the Object's Get Healed Cursor when targeting me if I have "GetHealedCursorName" configured. Defaults to: No.
- Added "UseMyGetRepairAtCursor". Uses my Get Repair At Cursor for the Get Repair At Cursor instead of the Object's Get Repair At Cursor when targeting me if I have "GetRepairAtCursorName" configured. Defaults to: No.
- Added "UseMyEnterCursor". Uses my Enter Cursor for the Enter Cursor instead of the Object's Enter Cursor when targeting me if I have "EnterCursorName" configured. Defaults to: No.
- Added "UseMySalvageCursor". Uses my Salvage Cursor for the Salvage Cursor instead of the Object's Salvage Cursor when targeting me if I have "SalvageCursorName" configured. Defaults to: No.

- ObjectCreationList:

- Added various configuring Variables for Objects Spawned.

```
{ "InheritsHealth", INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_inherits
{ "InheritsAIStates",   INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_inhe
{ "InheritsStatuses",   INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_inhe
{ "InheritsDisabledType",   INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_
{ "InheritsSelection",  INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_inhe
{ "InheritsSelectionClearsGroup",   INI::parseBool, NULL, offsetof( GenericObjectCreationNu
{ "HealthInheritType",      INI::parseIndexList,        TheMaxHealthChangeTypeNames, offset
```

- Added "InheritsHealth". Enables the Spawned Object to inherit the Health from the Object that Triggers the Module. The Health Inherit Type configuration is configured with 'HealthInheritType' Variable. Defaults to: No.
- Added "InheritsAIStates". Enables the Spawned Object to inherit the AI States (Moving, Attacking, Constructing, Repairing) of the Object that Triggers the Module. Defaults to: No.

- Added "InheritsStatuses". Enables the Spawned Object to inherit the Statuses of the Object that Triggers the Module. Defaults to: No.
- Added "InheritsDisabledType". Enables the Spawned Object to inherit the DisabledTypes of the Object that Triggers the Module. Defaults to: No.
- Added "InheritsSelection". Adds the Object Spawned to Selection if the Object that Triggers the Module is Selected. Defaults to: No.
- Added "InheritsSelectionClearsGroup". If the Object that Triggers the Module is Selected, clears the Selection Group. Defaults to: No
- Added "HealthInheritType". Determines the Health Inherit Type for “InheritsHealth” of the Object that Triggers the Module. Different Configuration Types:
  - PRESERVE_RATIO : Changes the Health according to the Health Ratio.
  - SAME_CURRENTHEALTH : Changes the Health to Match the Object’s Current Health.
  - ADD_CURRENT_DAMAGE : Changes the Missing Health to the Object’s Current Missing Health (Can kill if Missing Health exceeds Max Health.)
  - ADD_CURRENT_DAMAGE_NON_LETHAL: Changes the Missing Health to the Object’s Current Missing Health (If Missing Health exceeds Max Health, Object’s Health is set to 1.)
  - Defaults to: SAME_CURRENTHEALTH.

```
{ "TransferBombs",  INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_transferBombs )
{ "TransferAttackers",  INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_transferAtt
{ "TransferHijackers",  INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_transferHij
{ "TransferEquippers",  INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_transferEqu
{ "TransferParasites",  INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_transferPar
{ "TransferPassengers", INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_transferPas
{ "TransferToAssaultTransport", INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_tra
{ "TransferShieldedTargets",    INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_tra
{ "TransferShieldingTargets",   INI::parseBool, NULL, offsetof( GenericObjectCreationNugget, m_tra
```

- Added "TransferBombs". If Set to ‘Yes’, transfer any Sticky Bombs from the Object that Triggers the Module unto the Spawned Object. Defaults to: No.
- Added "TransferAttackers". If Set to ‘Yes’, transfer any Attackers from the Object that Triggers the Module unto the Spawned Object. Defaults to: No.
- Added "TransferHijackers". If Set to ‘Yes’, transfer any Hijackers from the Object that Triggers the Module unto the Spawned Object. Defaults to: No.

- Added "TransferEquippers". If Set to ‘Yes’, transfer any Equippers, excluding Parasites, from the Object that Triggers the Module unto the Spawned Object. Defaults to: No.
- Added "TransferParasites". If Set to ‘Yes’, transfer any Parasites from the Object that Triggers the Module unto the Spawned Object. Defaults to: No.
- Added "TransferPassengers". If Set to ‘Yes’, transfer any Passengers from the Object that Triggers the Module unto the Spawned Object. Defaults to: Yes
- Added "TransferToAssaultTransport". If Set to ‘Yes’, transfer the Object’s Assault Transport status onto the Spawned Object and Remove it from Assault Transport. Defaults to: No.
- Added "TransferShieldedTargets". If Set to ‘Yes’, transfer the Spawner’s SHIELDED_TARGET Status to the Spawned Objects along with the Shielder’s information and ProtectionTypes. Status and Information is then removed from the Spawner. Defaults to: No.
- Added "TransferShieldingTargets". If Set to ‘Yes’, transfer the Spawner’s Current Shielding Target’s Information to the Spawned Objects. Shielding Information is then removed from the Spawner. Defaults to: No.

AIUpdates:

General:

- Several AIUpdates have been reworked to be Sleepy. Previously AIUpdates would run every frame with/without commands, some are also not optimized and would be terrible for performance.
- These AIUpdate modules have been optimized to be Sleepy:
  - AssaultTransportAIUpdate
  - DeliverPayloadAIUpdate
  - DeployStyleAIUpdate
  - DozerAIUpdate
  - HackInternetAIUpdate
  - JetAIUpdate
  - MissileAIUpdate
  - WanderAIUpdate
  - WorkerAIUpdate

- AssaultTransportAIUpdate:

- AssaultTransportAIUpdate would bug out when used with PassengersAllowedToFire set to 'Yes'. It has been modified with codes from TransportAIUpdate to allow it to work compatible with PassengersAllowedToFire.
- If PassengersAllowedToFire is set to 'Yes', the AI will disable the behavior of ordering Passengers to Exit and Fire at the target and use the default TransportAIUpdate Open-Fire behavior.

```
{ "MembersGetHealedAtLifeRatio",                    INI::parseReal, NULL, offset  class AssaultT
{ "MembersRetreatOnWounded",                      INI::parseBool, NULL, offsetof( AssaultTranspor
{ "PassengersCanEnterOnMembersExit",                    INI::parseBool, NULL, offsetof( Assault
```

- Added MembersRetreatOnWounded. This was originally a feature of MembersGetHealedAtLifeRatio, which would result in the unit of the Assault Transport retreating back to the Transport when its health reaches the 'MembersGetHealedAtLifeRatio' threshold. This feature has been modified to allow the member to stay and fight if this Variable is configured to 'No'. Defaults to: Yes.
- Added PassengersCanEnterOnMembersExit. When an Assault Transports deploys its members they are considered exiting their Transport, allowing new Infantry to be able to head into the transport. If this is set to 'No', then members that are deployed by the Assault Transport to attack are counted towards the

transport count as if they are within Transport. Units that are deployed by Assault Transport are not affected by this matter and are able to Enter the Assault Transport by Command. Defaults to: Yes.

- DeployStyleAIUpdate:

- Behavior priority has been adjusted to check if the Object is about to move first before checking if the Object is about to attack. This is an attempt to fix for units with DeployStyleAIUpdate for being able to nudge ahead or move after Deploying if they are attacking.

```
{ "StatusToDeploy",     ObjectStatusMaskType::parseFromINI, NULL, offsetof( DeployStyleAIUpdateModuleDa
{ "StatusToUndeploy",       ObjectStatusMaskType::parseFromINI, NULL, offsetof( DeployStyleAIUpdateModu
{ "CustomStatusToDeploy",   INI::parseAsciiStringVector, NULL,  offsetof( DeployStyleAIUpdateModuleData
{ "CustomStatusToUndeploy", INI::parseAsciiStringVector, NULL,  offsetof( DeployStyleAIUpdateModuleData
{ "RemoveStatusAfterTrigger",   INI::parseBool, NULL,   offsetof( DeployStyleAIUpdateModuleData, m_remo
{ "DeployNoRelocate",   INI::parseBool, NULL,   offsetof( DeployStyleAIUpdateModuleData, m_deployNoRelo
{ "MoveAfterDeploy",    INI::parseBool, NULL,   offsetof( DeployStyleAIUpdateModuleData, m_moveAfterDep
{ "DeployNoRelocateUpgrades",   INI::parseAsciiStringVector, NULL,  offsetof( DeployStyleAIUpdateModule
{ "MoveAfterDeployUpgrades",    INI::parseAsciiStringVector, NULL,  offsetof( DeployStyleAIUpdateModule
{ "DeployInitiallyDisabled",    INI::parseBool, NULL,   offsetof( DeployStyleAIUpdateModuleData, m_depl
{ "ChangeInitiallyDisabledUpgrades", INI::parseDeployFunctionChangeUpgrade, NULL, offsetof( DeployStyle
```

- Added StatusToDeploy. Can now trigger Deploying if the Object has a Status present instead of needing the Object to Attack. Defaults to: None.
- Added StatusToUndeploy. Can now trigger Undeploying if the Object has a Status present instead of needing the Object to Move. Defaults to: None.
- Added CustomStatusToDeploy, and CustomStatusToUndeploy. Enable Support for Custom Status Types. Defaults to: None.
- Added RemoveStatusAfterTrigger. If a Status is Present for StatusToDeploy, the Object cannot trigger Undeploy by moving, vice-versa for StatusToUndeploy. This enables the Status to be cleared after the Unit has finished its Deploying or Undeploying Sequence. Defaults to: No.
- Added DeployNoRelocate. Disables Object from Undeploying when telling the Object to Move after Deploying. Can still be undeployed by using StatusToUndeploy. Defaults to: No.
- Added MoveAfterDeploy. Enables Objects to Move after Deploying. Requires DeployNoRelocate set to ‘Yes’ to function correctly. Defaults to: No.
- Added DeployNoRelocateUpgrades, and MoveAfterDeployUpgrades. DeployNoRelocate and MoveAfterDeploy can be override to ‘Yes’ or ‘No’ when certain Upgrades are present. Can be configured to be Triggered by More than One Upgrades, but will always check from first to last in order. Defaults to: Not Configured.

Example, declaring:

DeployNoRelocateUpgrades = Upgrade_GLABombTruckBioBomb No
Upgrade_GLABombTruckHighExplosiveBomb Yes
MoveAfterDeployUpgrades = Upgrade_GLABombTruckHighExplosiveBomb Yes

Will override the default DeployNoRelocate and MoveAfterDeploy status after the Object has been Upgraded with the Above Upgrades. If the Object is Upgraded with both Upgrade_GLABombTruckBioBomb and Upgrade_GLABombTruckHighExplosiveBomb, it will first check for Upgrade_GLABombTruckBioBomb to determine its configuration.

- Added DeployInitiallyDisabled. If set to 'Yes', the Object does not Deploy when trying to Attack, instead it can use its Weapons if the Turret's InitiallyDisabled is set to 'No'. Can still be Deployed by using StatusToDeploy. Defaults to: No.
- Added ChangeInitiallyDisabledUpgrades. Can override the Status of DeployInitiallyDisabled, TurretsFunctionOnlyWhenDeployed and Turret's InitiallyDisabled with Upgrades. Can be configured to be Triggered by More than One Upgrades, but will always check from first to last in order. Defaults to: Not Configured.

  To distinguish each variable, DeployInitiallyDisabled is configured with OBJECT, TurretsFunctionOnlyWhenDeployed is configured with TURRET and Turret's InitiallyDisabled is configured with DEPLOYED.

Example, declaring:

ChangeInitiallyDisabledUpgrades = OBJECT
Upgrade_GLABombTruckHighExplosiveBomb Yes TURRET
Upgrade_GLABombTruckHighExplosiveBomb No

ChangeInitiallyDisabledUpgrades = OBJECT Upgrade_GLABombTruckBioBomb No TURRET Upgrade_GLABombTruckBioBomb Yes

ChangeInitiallyDisabledUpgrades = DEPLOYED
Upgrade_GLABombTruckBioBomb No

Will override the respective values when the Object has been Upgraded with the declared Upgrades. Remember, orders are checked from first to last.

- MissileAIUpdate:

```
{ "AllowRetargeting", INI::parseBool, NULL, offsetof(MissileAIUpdateModuleData,
```

- Added AllowRetargeting. Normally when a Projectile with “TryToFollowTarget” set to ‘Yes’ loses its target, it will self detonate. Setting this to ‘Yes’ will cause it to find a new target within the Projectiles current range. Does nothing if “TryToFollowTarget” is not set to ‘Yes’. Defaults to: No.

BehaviorModules:

- BattlePlanBonusBehavior (Modding Devs' Implemented Feature):

```
{ "CustomWeaponBonus",  parseBPCustomWeaponBonus, NULL, 0 },
{ "CustomStatusToSet",  parseBPCustomStatusToSet, NULL, 0 },
{ "CustomStatusToClear",    parseBPCustomStatusToClear, NULL, 0 },
```

- Added CustomWeaponBonus, CustomStatusToSet, and CustomStatusToClear. Enable Custom Statuses to work with StealthUpdate. Enable Custom Weapon Bonuses and Custom Status Types to work with the implemented feature.

CollideModules:

- FireWeaponCollide:

```
{ "CollideWeapon",      INI::parseWeaponTemplate,                   NULL, offsetof( FireWeaponC
{ "FireOnce",               INI::parseBool,                                 NULL, offse
{ "RequiredStatus",     ObjectStatusMaskType::parseFromINI, NULL, offsetof( FireWeaponCollideModule
{ "ForbiddenStatus",    ObjectStatusMaskType::parseFromINI, NULL, offsetof( FireWeaponCollideModule
{ "RequiredCustomStatus",       INI::parseAsciiStringVector,    NULL, offsetof( FireWeaponCollideMo
{ "ForbiddenCustomStatus",  INI::parseAsciiStringVector,    NULL, offsetof( FireWeaponCollideModule
```

- Added RequiredCustomStatus, and ForbiddenCustomStatus. Enable Custom Statuses to work with FireWeaponCollide. Functions the same way existing RequiredStatus and ForbiddenStatus functions.
- Now respects the Weapon Template's Upgrade and Status requirements when checking whether it should fire.

- AdvancedCollide (Modding Devs' Implemented Feature):

```
{ "RequiredCustomStatus",   INI::parseAsciiStringVector, NULL,  offsetof( AdvancedCollideModuleData
{ "ForbiddenCustomStatus",  INI::parseAsciiStringVector, NULL,  offsetof( AdvancedCollideModuleData
{ "TargetRequiredCustomStatus", INI::parseAsciiStringVector, NULL,  offsetof( AdvancedCollideModule
{ "TargetForbiddenCustomStatus",    INI::parseAsciiStringVector, NULL,  offsetof( AdvancedCollideMo
```

- Added RequiredCustomStatus, ForbiddenCustomStatus, TargetRequiredCustomStatus, and TargeForbiddenCustomStatus. Enable Custom Statuses to work with AdvancedCollide. Functions the same way existing RequiredStatus and ForbiddenStatus functions.
- Now respects the Weapon Template's Upgrade and Status requirements when checking its "CollideWeapon" to see whether it should fire.

ContainModules:

General:

- The amount of ContainMax or Slots for TransportContain can now be modified with Upgrades. The new amount of ContainMax only affects the Logic of Units that are Not currently contained inside the Object. Currently Contained Units are not affected and only counts towards the Containers Unit counts when determining whether an Object can Enter into the Container.
- All Contain modules, except TunnelContain and CaveContain, are implemented with this new feature.

```
{ "ContainMaxTriggeredBy",          INI::parseAsciiStringWithColonVectorAppend, NULL, offsetof
{ "ContainMaxConflictsWith",        INI::parseAsciiStringWithColonVectorAppend, NULL, offsetof
{ "ContainMaxRequiresAllTriggers",  INI::parseIntVector, NULL, offsetof( OpenContainModuleData
```

- Added ContainMaxTriggeredBy. Configure by Registering first the new Value to replace the current ContainMax, then write down the Upgrades that are able to Trigger it. If an Upgrade is present, the Module will find the approximate Value that is tied to the Upgrade. If multiple Upgrades with different Values are present, the Module will select the Highest Value for its ContainMax. Subsequent values declared after an Upgrade but without any later Upgrades are ignored. Declaration format is demonstrated below. Defaults to: Not configured.

  Example:

  ContainMaxTriggeredBy = 11:Upgrade_ChinaAssaultTransportExtraSlotUpgrade 50:Upgrade_ChinaFullForce Upgrade_ChinaOverlordBattleBunker 51 52

  When Upgraded with Upgrade_ChinaAssaultTransportExtraSlotUpgrade, the ContainMax increased from 10 to 11. However, when Upgrade_ChinaFullForce upgrade is present, the ContainMax changes to 50. When Both Upgrades are Present, the value 50 is taken because it is the highest. Value 51, and 52 are ignored because no subsequent Upgrades are present.

- Added ContainMaxConflictsWith. Configured the same way as ContainMaxTriggeredBy. If an Upgrade is present, the Module will find the approximate Value that is tied to the Upgrade. That Upgrade will initiate the ‘ConflictsWith’ Upgrade for the tied Value, skipping the Value to be considered for Upgrade. Defaults to: Not configured.

  Example:

ContainMaxConflictsWith = 11:Upgrade_ChinaChainGuns 15:Upgrade_ChinaBlackNapalm

If Upgrade_ChinaChainGuns is present, while the Object has Upgrade_ChinaAssaultTransportExtraSlotUpgrade upgraded, the Slots will not be increased from 10 to 11, but if Upgrade_ChinaFullForce is present while Upgrade_ChinaBlackNapalm is present, the Slots will still be increased from 10 to 50 because Upgrade_ChinaBlackNapalm only Conflicts with Upgrades that are tied to a Value of 15.

- Added ContainMaxRequiresAllTriggers. Declared in an array of Numbers. Numbers that match the Value of ContainMaxTriggeredBy Upgrades Requires all of their Upgrades to be Present to trigger the Value. Defaults to: Not configured.

  Example:

  ContainMaxRequiresAllTriggers = 11 50

  That means both Values declared under ‘ContainMaxTriggeredBy‘ require All Upgrades present to trigger the Upgrade. The value of 11 only requires only Upgrade_ChinaAssaultTransportExtraSlotUpgrade to be present, while the value of 50 requires both Upgrade_ChinaFullForce and Upgrade_ChinaOverlordBattleBunker to be present for passing the Upgrade.

- RiderChangeContain:

- Memory Pool allocated increased from { 128, 32 } to { 512, 128 }.
- Added CustomStatusType support for Rider Parsing System. When registering the StatusType for a Rider, the system will first consider whether that is a valid StatusType within a system. If yes, it will register it as a StatusType. If not, it will register as a CustomStatusType.
- Expanded RiderChangeContain to be able to register more than 8 RiderTemplates and able to use different means of changing their Templates other than just switching Riders.
- Expanded RiderChangeContain to work with Transport modules. The default RiderChangeContain Behavior is configured to Destroy itself whenever a Passenger has exited, and doesn’t have many workarounds to make it a suitable Transport. The module has been expanded to be able to work as an IFV with Transport with the expanded modules.

- Added RiderCustom expansion for RiderChangeContain, enabling an Object to go beyond just 8 riders. RiderCustom are declared the same format as any of the riders. The declaration is appending, that means to declare a new rider, just use the same 'RiderCustom' function.

Example, just by declaring new 'RiderCustom', it will automatically register into the unit's database:

RiderCustom = Demo_GLAInfantryNewInfantry REALLYDAMAGED WEAPON_CARBOMB STATUS_CUSTOMRIDER GLAVehicleCombatBikeDefaultCommandSet SET_SLUGGISH

RiderCustom = Demo_GLAInfantryNewHero TOPPLED WEAPONSET_MINE_CLEARING_DETAIL STATUS_RIDER9999 GLAVehicleCombatBikeDefaultCommandSet SET_SLUGGISH

- Added RiderChangeOnStatusTypes. If set to 'Yes', enables the Object to simply switch their Rider Templates based on the StatusType. This does not mean that the Object does not require a rider, but that the template can be changed with just StatusTypes alone.
- If the Object changed Riders, it will function accordingly to RiderChangeContain, switching the StatusType to the Rider's StatusType. The Rider Template can still be changed by changing the StatusTypes.
- The feature ~~has a Time based check with a few checks and might cost a small amount of performance.~~ (This has been updated) checks whenever a new Status is granted or removed from the Object. Fully compatible with DamageStatusType that only grants statuses temporarily. After the status expires, it will revert back to the last Rider Template. This feature respects all the timer of statuses given, and will revert back to a state with or without the appropriate Statuses.
- Added RiderUseUpgradeNames. If set to 'Yes', Riders can change their modules after an upgrade has been completed if the upgrade is registered in place of the rider name.
- Removing Upgrades will now set the Rider to the last available Template.

For Example,

Object CustomCombatBike

RiderCustom = Upgrade_GLAAPRockets RIDER3 WEAPON_RIDER3 STATUS_RIDER3 GLAVehicleCombatBikeDefaultCommandSet SET_NORMAL

Rider5 = Upgrade_ChinaOverlordPropagandaTower RIDER5 WEAPON_RIDER5 STATUS_RIDER5 GLAVehicleCombatBikeDefaultCommandSet SET_SLUGGISH

End

- When Upgrade_GLAAPRockets is Upgraded, it will change all the Template into RiderCustom, while after upgrading it with Upgrade_ChinaOverlordPropagandaTower will change the template to Rider5.
- On Order, the Rider will Consider the Most Recently pushed Template to use. For example, after triggering Upgrade_GLAAPRockets, the Rider Template will switch to the ones declared by the Upgrade. Then after the Object is upgraded with Upgrade_ChinaOverlordPropagandaTower, it will switch to Rider5 Template. If then applied with STATUS_RIDER3 by a weapon or Status Bits Upgrade, it will switch to the template of Upgrade_GLAAPRockets. Then if the Object is Occupied with Jarmen Kell, it will then switch to that Rider Template.

```
{ "DontDestroyPassengersOnKill",    INI::parseBool,            NULL, offsetof( RiderChangeContainModule
{ "ScuttleRequirements",        INI::parseIndexList,           TheScuttleNames, offsetof( RiderChangeCo
{ "DontEvacuateOnEnter",        INI::parseBool,                NULL, offsetof( RiderChangeContainModule
{ "CanContainNonRiders",        INI::parseBool,                NULL, offsetof( RiderChangeContainModule
{ "MoreThanOneRiders",      INI::parseBool,              NULL, offsetof( RiderChangeContainModuleData
```

- Added DontDestroyPassengersOnKill. RiderChangeContain by default completely removes the Passenger from the game if it was destroyed even when configured with no DamagePercentToUnits. If set to ‘Yes’, this disables it. Defaults to: No.
- Added ScuttleRequirements. If a Contained Object has been removed from an Object with RiderChangeContain, the Bike will ‘Scuttle’, and be destroyed later on. This has been separated into 3 settings, ‘ON_EXIT’, ‘ON_NO_PASSENGERS’ and ‘NEVER’. ON_EXIT is the default behavior, destroying itself whenever a Passenger has been removed by while no Passengers are Entering. ‘ON_NO_PASSENGERS’ triggers the same behavior as ‘ON_EXIT’, but only when the Unit has no Passengers after exiting. ‘NEVER” means the Object will never scuttle from removing Passengers. Defaults to: ON_EXIT.
- Added DontEvacuateOnEnter. When a Passenger has entered a Unit with RiderChangeContain, it will forcefully Evacuate whatever is Contained, leaving only the recent Passenger in it. Setting this to ‘Yes’ disables this feature. Defaults to: No.
- Added CanContainNonRiders. Added support for Containing non-Riders for units that use RiderChangeContain module. They work similarly to Transports and do

not affect the Rider Templates in any way. The difference being Riders can enter on a fully Contained Unit with RiderChangeContain while Non-riders cannot. When a Rider enters a fully Contained module, it will start to remove the last Passenger from the Contain until it has a suitable Transport Slot and then enter it. Defaults to: No.

- Added MoreThanOneRiders. Since RiderChangeContain supports Transport Modules, you can now load more than one Riders, but they won't change the existing Rider Template. Setting this to 'Yes' changes the RiderTemplate to the latest Rider after Containing it. Defaults to: No.
- Added ways for Riders to Switch Templates if supposedly the current Rider that uses the Template has exited the Contain. If MoreThanOneRiders is set to 'No', the Object will register the first or earliest Rider for its Template. If MoreThanOneRiders is set to 'Yes', the Object will register the latest Rider for its Rider Template.
- If a Rider has no Rider Template, it will consider Rider1 for its Template, though no Rider Template can be overridden by any Newer Template, such as RiderChangeOnStatusTypes and RiderUseUpgradeNames, thereafter.
- Evacuation for RiderChange has been changed, it now exits the last Passenger to the first, except when evacuating via Containing. This design works better with the newly implemented RiderChangeContain feature.

- TunnelContain:

```
{ "AllowInsideKindOf",              KindOfMaskType::parseFromINI, NULL, offsetof( OpenContainModuleData
{ "ForbidInsideKindOf",             KindOfMaskType::parseFromINI, NULL, offsetof( OpenContainModuleData
{ "PassengersAllowedToFire",    INI::parseBool, NULL, offsetof( OpenContainModuleData, m_passengersAllo
```

```
{ "AllowAlliesInside",              INI::parseBool, NULL, offsetof( OpenContainModuleData,
{ "AllowEnemiesInside",             INI::parseBool, NULL, offsetof( OpenContainModuleData,
{ "AllowNeutralInside",             INI::parseBool, NULL, offsetof( OpenContainModuleData,
```

- Modified so that the above features now work with Tunnel Contain.
- PassengersAllowedToFire now works differently. It works by replacing the Tunnel Guard for the Tunnel. Normally when a Unit with TunnelContain is damaged, Units will exit and attack the Damager. However, with Fire Ports, the Units will Open Fire from the Tunnel onto the Object instead.
- If a Multiple Unit Exists with TunnelContain configured with PassengersAllowedToFire are attacked, the Unit that is most recently seen will have Contained Units Open Fire from within.
- This attack sequence can be overwritten by right-clicking. (Same as PassengersAllowedToFire.)

```
{ "RemoveOtherTunnelBunkerOnUpgrade", INI::parseBool, NULL, offsetof( TunnelContainModuleData, m_remove
{ "UpgradesDisableOtherTunnelGuard", INI::parseAsciiStringVector, NULL, offsetof( TunnelContainModuleDa
{ "UpgradesDisableOwnTunnelGuard", INI::parseAsciiStringVector, NULL, offsetof( TunnelContainModuleData
{ "InitialPayload", parseInitialPayload, NULL, 0 },
```

- ~~Added UpgradesToTriggerBunker. Which states any of the Upgrades required to trigger PassengersAllowedToFire. Requires PassengersAllowedToFire set to 'Yes' to function. If the Object is not Upgraded with the declared Upgrade, it will use Tunnel Guard behavior. Defaults to: None.~~
  PassengersFireUpgrade is now compatible with TunnelContain. Works the same way as the features above.
- Added RemoveOtherTunnelBunkerOnUpgrade. Which when upgraded with ~~UpgradesToTriggerBunker~~ PassengersFireUpgrade, prevents all other Tunnels from Firing with PassengersAllowedToFire and also removes their Upgrades required to trigger PassengersFireUpgrade. Defaults to: No.
  NOTE: Do not use Player Upgrades for PassengersFireUpgrade when this feature is enabled, it will crash the game.
- Added UpgradesDisableOtherTunnelGuard. Prevents all other Tunnels for using Tunnel Guard except the current Unit. Works by declaring the Upgrades to Trigger the condition. Only requires one of the Upgrades present to trigger. After Upgraded, removes the Tunnel Guard Status and Upgrades of other Tunnels to trigger this feature. It works the same as the feature above, but for Tunnel Guard,

which exits Contained Units to fire at the Damager. Like above, do not Use Player Upgrades when this feature is enabled. Defaults to: None.

- Added UpgradesDisableOwnTunnelGuard. Prevents the Object itself from using Tunnel Guard by using an Upgrade. Works by declaring the Upgrades to Trigger the condition. Only requires one of the Upgrades present to trigger. Defaults to: None.
- Added InitialPayload. TunnelContain can now come with existing units contained within. If the Tunnel is full, the units will exit from the builded structure. Defaults to: Nothing.

- CaveContain:

- CaveContain is an existing feature that is a more dynamic feature of TunnelContain. A cave can be owned by default by entering into an empty Tunnel. However, the ownership is forfeit when the Tunnel is empty. You can declare separated Tunnels by using different CaveIndexes. Though it only works by declaring it to Neutral buildings, using this feature on your own buildings while trying to build them crashes the game.
- CaveContain has been modified so that it now works on your own buildings and your own teams. No longer crashes the game when trying to build your own Caves. Caves will also no longer convert to Neutral side, but instead back to their previous owning side after the last Unit has exited the Cave.
- CaveContain also has been made more versatile and compatible with features such as Capturing. When an Object with CaveContained is Captured, the ownership of the current Cave object will be permanently granted to the Capturer, and will not be shifted even if there are other player's units within. After Capturing, if there are any units Contained within the Object that is not owned by the player, they will be forcefully removed from the Cave.
- AllowEnemiesInside has also been set to 'No' by default for CaveContain, as Caves can now be captured and granted permanent ownership.
- Unlike TunnelContain, CaveContain is designed to work with Neutral Teams, so there's no Tunnel Guard Protection feature.

```
{ "AllowInsideKindOf",            KindOfMaskType::parseFromINI, NULL, offsetof( OpenContainModuleData
{ "ForbidInsideKindOf",           KindOfMaskType::parseFromINI, NULL, offsetof( OpenContainModuleData
```

```
{ "AllowAlliesInside",               INI::parseBool, NULL, offsetof( OpenContainModuleData,
{ "AllowEnemiesInside",              INI::parseBool, NULL, offsetof( OpenContainModuleData,
{ "AllowNeutralInside",              INI::parseBool, NULL, offsetof( OpenContainModuleData,
```

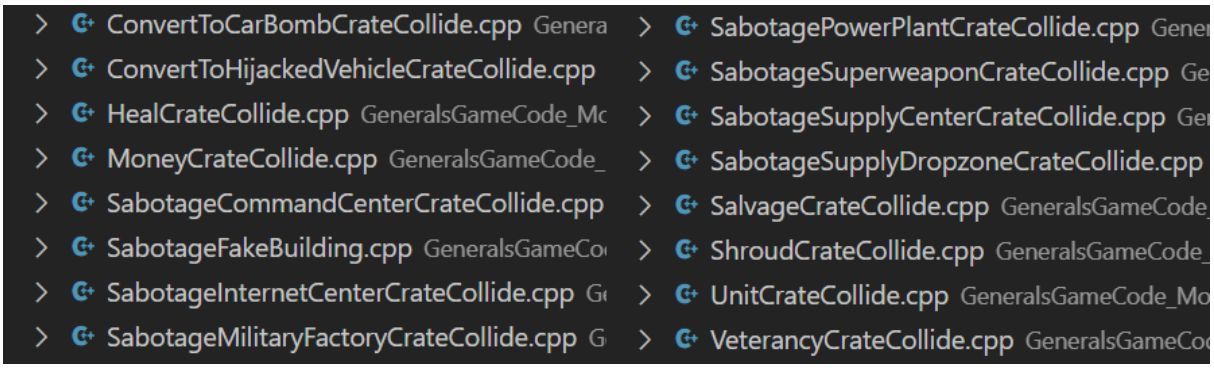
- Like TunnelContain, CaveContain has also been modified to allow the above features to be compatible with it. The only exception is PassengersAllowedToFire, which has not been implemented.

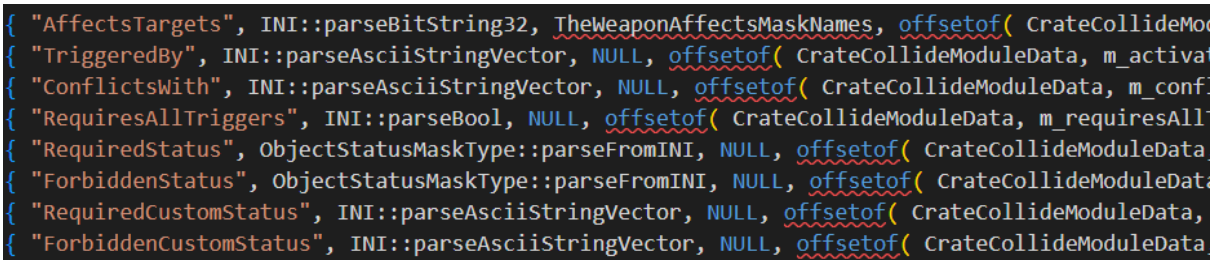
```
{ "CaveIndex", INI::parseInt, NULL, offsetof( CaveContainModuleData, m_caveIndexData ) },
{ "CaveHasOwner", INI::parseBool, NULL, offsetof( CaveContainModuleData, m_caveHasOwner ) },
{ "CaveUsesTeams", INI::parseBool, NULL, offsetof( CaveContainModuleData, m_caveUsesTeams ) }
{ "CaveCaptureLinkCaves", INI::parseBool, NULL, offsetof( CaveContainModuleData, m_caveCaptur
{ "InitialPayload", parseInitialPayload, NULL, 0 },
```

- Added CaveHasOwner. Normally when an Object enters an empty Cave, it converts all the Caves that use the same CaveIndex onto the side of the player that controls the Object. This feature disables that for the Object. However, other player's Units can still be found in any Caves with the same CaveIndex. Defaults to: No.
- Added CaveUsesTeams. From the example above, we know CaveContain doesn't respect teams if there are multiple teams using the same CaveIndexes. Their Units will get mixed up in the Cave. This feature is designed so that each player can have their own separated Cave pathways while using the same CaveIndexes. Caves with CaveHasTeams set to 'Yes' also prevent Units from being Forcefully Exited after Captured, unless it is the last Cave present. Useful for making faction buildable Caves or Tech Caves. Defaults to: No.
- Added CaveCaptureLinkCaves. After revamping, only the current Cave that is Captured will defect onto the Captured side. This feature is enabled so that when captured, all Caves with the same CaveIndexes will defect onto the Captured Side. If CaveHasTeams is set to 'Yes', all the Caves with the same CaveIndex of that Team will be converted onto the Capturer's side. Defaults to: No.
- Added InitialPayload. CaveContain can now come with existing units contained within. If a Cave is full, the units will exit from the builded structure. Defaults to: Nothing.

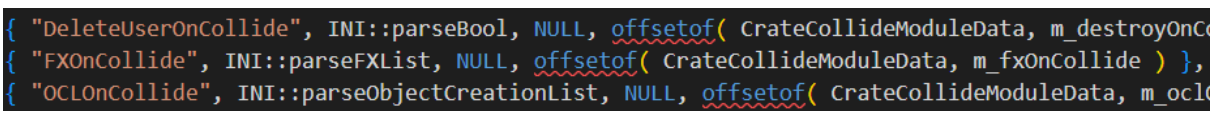
CrateCollideModules:

- List of Modules that is a CrateCollide Module:



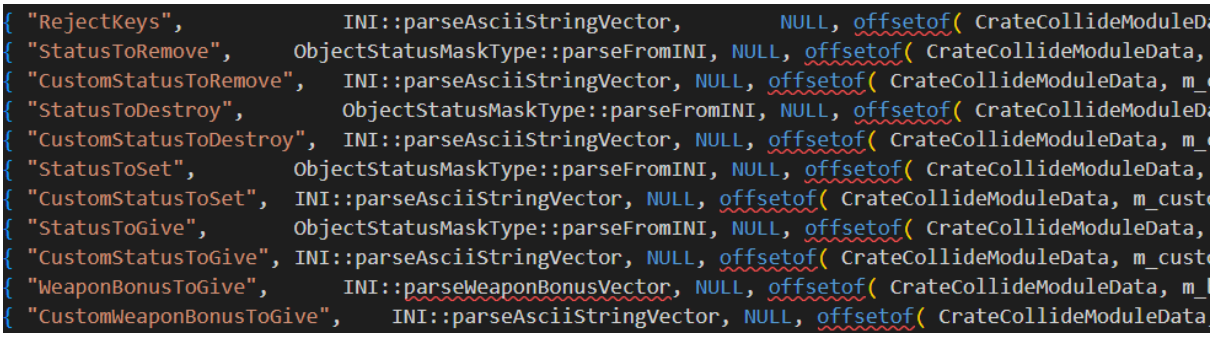


- Added AffectsTargets. Configured with ALLIES, ENEMIES, and NEUTRALS. Allow only specific sides to be able to Collide with this Module. Defaults to all if not configured. Defaults to: Not configured.
- Added TriggeredBy, ConflictsWith, and RequiresAllTriggers. They are assigned the same way as Upgrade Modules. If assigned, they check whether the Collider meets the Upgrade criteria to Collide. Defaults to: None.
- Added RequiredStatus, and ForbiddenStatus. They are assigned the same way as Behavior Modules. If assigned, they check whether the Collider meets the Status criteria to Collide. Defaults to: None.
  NOTE: RequiredStatus checks for All of the declared Statuses while ForbiddenStatus checks for Any of the declared Statuses.
- Added RequiredCustomStatus, and ForbiddenCustomStatus to support CustomStatusTypes.



- Added DeleteUserOnCollide. If set to ‘Yes’, it will delete the object after triggering its CrateCollide abilities.

- It uses the same form of Deletion used by DeletionUpdate, which would not inform or alert the system on delete.
- Added FXOnCollide and OCLOnCollide. If set, spawns the respective FX and OCL after triggering its CrateCollide abilities.



> These modules above are configured once the Collider has collided with the Object, but several Collide Modules that work together with HijackerUpdate or OpenContain Modules with EquipCrateCollide are able to revert the Collide properties. Properties are reverted when the Collider is removed or destroyed from the Object.

> These modules include:

- ConvertToCarBombCrateCollide
- ConvertToHijackedVehicleCrateCollide
- EquipCrateCollide

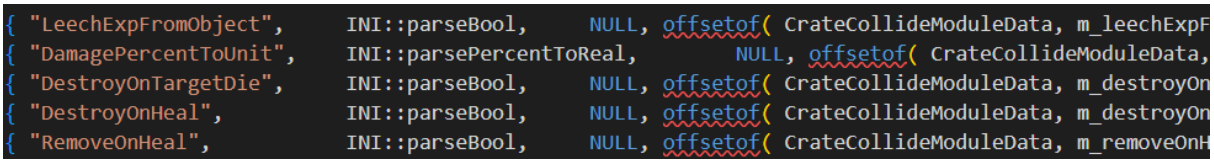
> ConvertToCarBombCrateCollide has been modified to be able to work compatibly with HijackerUpdate. (See HijackerUpdate for more details.)

> ConvertToHijackedVehicleCrateCollide has been modified for the Hijacker to be able to be removed or destroyed from the Hijacked Vehicle. (See HijackerUpdate Below for more details.)

- Added RejectKeys. Registers the Reject Keys for the Target Object after Colliding with the Object. Colliders that have Any of the Reject Keys that are registered on the Target cannot Collide. When a Collider is removed from the Object, the Respective Reject Keys are also removed. An Object can be registered with more than one Reject Key. Can be configured with more than one Reject Keys. Defaults to: Not configured.
- Added StatusToRemove, and CustomStatusToRemove. If the Object has the Status or Custom Status while the Collider with HijackerUpdate configured has already collided with the Object, the Collider is immediately removed or ejected

from the Object. Requires the Collider to have HijackerUpdate Module configured to enable this feature. Defaults to: None.

- Added StatusToDestroy, and CustomStatusToDestroy. If the Object has the Status or Custom Status while the Collider with HijackerUpdate configured has already collided with the Object, the Collider is immediately destroyed and the Object reverts the properties that are applied by the Collider. Requires the Collider to have HijackerUpdate Module configured to enable this feature. Defaults to: None.
- Added StatusToSet, and CustomStatusToSet. Set the Statuses of the Collider when Colliding with an Object. If the Collider is removed, then the Respective Statuses are also removed from the Collider. Defaults to: None.
- Added StatusToGive, and CustomStatusToGive. Set the Statuses of the Object that the Collider Collides with. If the Collider is removed or destroyed, then the Respective Statuses are also removed from the Object. Defaults to: None.
- Added WeaponBonusToGive, and CustomWeaponBonusToGive. Set the Weapon Bonuses for the Object that the Collider Collides with. If the Collider is removed or destroyed, then the Respective Weapon Bonuses are also removed from the Object. Defaults to: None.



- Added LeechExpFromObject. If a Collider with HijackerUpdate has collided with the Object, it will set the Object according to its Veterancy if the Collider has a higher veterancy. Defaults to: Yes for ConvertToHijackedVehicleCrateCollide; while No for ConvertToCarBombCrateCollide and EquipCrateCollide.
- Added DamagePercentToUnit. Collider with HijackerUpdate can be configured to receive a portion of damage from the Collided Object when it receives any damage, and it can be killed by this mechanic. Killed Colliders are considered removed from the Collided Object. Defaults to: 0%.
- Added DestroyOnTargetDie. If the Unit has a Collider with HijackerUpdate attached onto it, the Collider is destroyed if the Unit has Died. Defaults to: No for ConvertToHijackedVehicleCrateCollide and EquipCrateCollide; while Yes to ConvertToCarBombCrateCollide.

- Added DestroyOnHeal. If the Unit has a Collider with HijackerUpdate attached onto it, the Collider is destroyed if the Unit has gained Health in any way. Defaults to: No.
- Added RemoveOnHeal. If the Unit has a Collider with HijackerUpdate attached onto it, the Collider is removed if the Unit has gained Health in any way. Defaults to: No.

```
{ "CursorName", INI::parseAsciiString, NULL, offsetof( CrateCollideModuleData,
```

- Added CursorName. Adjust the Cursor when trying to Collide with the CrateCollide Module if configured. Defaults to: Not configured.

- EquipCrateCollide (New Module!):

- Memory Pool allocated with { 256, 128 }.
- A new Collide module that latches the Collider onto an Object, effectively 'equipping' it. While equipped, properties of CrateCollideModuleData are applied respectively on the Collider and the Object.
- Can also be configured alongside the HijackerUpdate module to enable the Collider to be "ejected" from the Object if the equipped Object has been killed. It also enables the properties set by CrateCollideModuleData to be reverted by StatusToRemove or StatusToDestroy.
- FireWeaponUpdate, FireWeaponAdvancedUpdate and FireWeaponWhenDamagedBehavior are also triggered on the Collided Object's position after equipping.

```
{ "IsContain",              INI::parseBool,     NULL, offsetof( EquipCrateCollideModuleData,
{ "IsParasite",             INI::parseBool,     NULL, offsetof( EquipCrateCollideModuleData,
{ "IsUnique",               INI::parseBool,     NULL, offsetof( EquipCrateCollideModuleData,
{ "ParasiteKey",            INI::parseAsciiString,      NULL, offsetof( EquipCrateCollideModu
{ "CanPassiveAcquire",      INI::parseBool,     NULL, offsetof( EquipCrateCollideModuleData,
{ "DestroyOnClear",         INI::parseBool,     NULL, offsetof( EquipCrateCollideModuleData,
```

- Added IsContain. If set to 'Yes', EquipCrateCollide will consider the Objects' Contain Modules for equipping and applying properties of CrateCollideModuleData towards the Object. If an Equipped Object with IsContain set to 'Yes' exits from the Object's Contain, the properties of CrateCollideModuleData are reverted after exiting. Defaults to: No.
- Added IsParasite. Uses a different approach for EquipCrateCollide. If set to 'Yes', this module can no longer voluntarily target Allies and if the Object has a Weapon that can attack the Collidable Object, it can no longer target to Collide

with the Object. The module can only be applied through GUI Commands or while the Object is attacking the Collidable Object, while it is within Collidable distance.
If set to 'Yes', it can also be 'Cleared' with Healing Modules. When Parasites are cleared, they are Removed from the Collided Object and will revert the properties applied from CrateCollideModuleData.
Defaults to: No.

- Added features for modules to clear Parasites:

> AutoHealBehavior:

```
{ "ClearsParasite",             INI::parseBool,
{ "ClearsParasiteKeys",         INI::parseAsciiStringVector,
```

- ClearsParasite - Does Healing from this Module clears Parasites? Defaults To: No.
- ClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> DozerAIUpdate

```
{ "RepairClearsParasite",           INI::parseBool, NULL, offsetof( DozerAIUpdate
{ "RepairClearsParasiteKeys",       INI::parseAsciiStringVector, NULL, offsetof(
```

- RepairClearsParasite - Does Repair from this Module clears Parasites? Defaults To: Yes.
- RepairClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> WorkerAIUpdate

```
{ "RepairClearsParasite",           INI::parseBool, NULL, offsetof( WorkerAIUpdate
{ "RepairClearsParasiteKeys",       INI::parseAsciiStringVector, NULL, offsetof( W
```

- RepairClearsParasite - Does Repair from this Module clears Parasites? Defaults To: Yes.
- RepairClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> BaseRegenerateUpdate

```
{ "BaseRegenClearsParasite", INI::parseBool, NULL,  offsetof( BaseRegenerateUpdate
{ "BaseRegenClearsParasiteKeys", INI::parseAsciiStringVector, NULL, offsetof( Base
```

- BaseRegenClearsParasite - Does Regeneration from this Module clears Parasites? Defaults To: No.
- BaseRegenClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> FlightDeckBehavior

```
{ "HealingClearsParasite",          INI::parseBool, NULL, offsetof( FlightDeckBehavior
{ "HealingClearsParasiteKeys",      INI::parseAsciiStringVector, NULL, offsetof( Fligh
```

- HealingClearsParasite - Does Healing from this Module clears Parasites? Defaults To: No.
- HealingClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> GarrisonContain

```
{ "HealingClearsParasite",          INI::parseBool, NULL, offsetof( GarrisonContain
{ "HealingClearsParasiteKeys",      INI::parseAsciiStringVector, NULL, offsetof( Ga
```

- HealingClearsParasite - Does Healing from this Module clears Parasites? Defaults To: No.
- HealingClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> ParkingPlaceBehavior

```
{ "HealingClearsParasite",          INI::parseBool, NULL, offsetof( ParkingPlaceBehavior
{ "HealingClearsParasiteKeys",      INI::parseAsciiStringVector, NULL, offsetof( Parking
```

- HealingClearsParasite - Does Healing from this Module clears Parasites? Defaults To: No.
- HealingClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> PropagandaTowerBehavior

```
{ "HealingClearsParasite",          INI::parseBool, NULL, offsetof( PropagandaTowerBehavior
{ "HealingClearsParasiteKeys",      INI::parseAsciiStringVector, NULL, offsetof( Propaganda
```

- HealingClearsParasite - Does Healing from this Module clears Parasites? Defaults To: No.
- HealingClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> RepairDockUpdate

```
{ "RepairClearsParasite", INI::parseBool, NULL, offsetof( RepairDockUpdate
{ "RepairClearsParasiteKeys", INI::parseAsciiStringVector, NULL, offsetof(
```

- RepairClearsParasite - Does Repair from this Module clears Parasites? Defaults To: Yes.
- RepairClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> SlavedUpdate

```
{ "RepairClearsParasite",    INI::parseBool, NULL, offsetof( SlavedUpdate
{ "RepairClearsParasiteKeys",    INI::parseAsciiStringVector, NULL, offse
```

- RepairClearsParasite - Does Repair from this Module clears Parasites? Defaults To: Yes.
- RepairClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> TeleporterAIUpdate (Modding Devs' Implemented Feature)

```
{ "TeleportClearsParasite", INI::parseBool, NULL, offsetof(TeleporterAIUpda
{ "TeleportClearsParasiteKey", INI::parseAsciiStringVector, NULL, offsetof(
```

- TeleportClearsParasite - Does Teleporting from this Module clears Parasites? Defaults To: Yes.
- TeleportClearsParasiteKey - The list of Parasite Keys from this Module. Defaults To: Not configured.

> Weapons (WeaponTemplates)

```
{ "ClearsParasite",                    INI::parseBool,          NULL,
{ "ClearsParasiteKeys",                INI::parseAsciiStringVector,
```

- ClearsParasite - Does Hitting Objects with this Weapon clears Parasites? Defaults To: No.
- ClearsParasiteKey - The list of Parasite Keys from this Weapon. Defaults To: Not configured.

- Added IsUnique. If set to 'Yes', this module cannot Collide with Objects that are already equipped. Defaults to: No.
- Added ParasiteKey. If configured with a Parasite Key, only the Healing Modules above that contain the matching Parasite Key with the Collided EquipCrateCollide Module can clear the Parasite. Defaults to: Not configured.
- Added CanPassiveAcquire. If set to 'Yes', the Collider can fire their weapons from the Equipped Object's Position after Equipping. Defaults to: No.
- ~~LeechExpFromObject. When configured with HijackerUpdate, does the Object gain the Collider's Experience Points after colliding like Pilots? Defaults to: No.~~ Moved to CrateCollideModule.
- ~~Added DestroyOnHeal. When configured with HijackerUpdate, does the Collider get Destroyed if the Equipped Object restores or gains Health? Defaults to: No.~~ Moved to CrateCollideModule.
- ~~Added RemoveOnHeal. When configured with HijackerUpdate, does the Collider get Removed if the Equipped Object restores or gains Health? Defaults to: No.~~ Moved to CrateCollideModule.
- Added DestroyOnClear. If IsParasite is set to 'Yes', and the Collider is Cleared from the above Modules, Collider is Destroyed instead of being Removed from the Object if set to 'Yes'. Defaults to: No.

- NOTE: This module also has its own Guard Mode like 'HijackGuard'. To declare them, declare 'EnterGuard = Yes' and one Guard Mode below:

- EquipGuard: When Guarding, the Object will search any Ally Equippable Objects nearby to Equip.
- ParasiteGuard: When Guarding and the Object has 'IsParasite = Yes' declared on its EquipCrateCollide module, the Object will search any Enemy Parasitable Objects nearby to latch onto.

- Added CursorName. Adjust the Cursor when trying to Collide with the CrateCollide Module if configured. Defaults to: Not configured.

- SalvageCrateCollide:

- Pickup features are de-hardcoded. Can now customize which KindOfs to be qualified for Salvaging Armor or Salvaging Weapons.
- Can now customize the level of features to obtain, and the maximum levels of each feature.
- Can now also customize the pickup order for each feature, not having to start from Armor Flags.
- Salvage Crate can now be configured with more than just existing features for Salvaging.

```
{ "RequiredKindOfWeaponSalvager", KindOfMaskType::parseFromINI, NULL, offsetof( SalvageCrat
{ "ForbiddenKindOfWeaponSalvager", KindOfMaskType::parseFromINI, NULL, offsetof( SalvageCra
{ "RequiredKindOfArmorSalvager", KindOfMaskType::parseFromINI, NULL, offsetof( SalvageCrate
{ "ForbiddenKindOfArmorSalvager", KindOfMaskType::parseFromINI, NULL, offsetof( SalvageCrat
```

- Added RequiredKindOfWeaponSalvager. Determine the Multiples KindOfs required to qualify for Salvaging Weapon Flags. Can be configured with NONE for no qualifications. Defaults to: WEAPON_SALVAGER.
- Added ForbiddenKindOfWeaponSalvager. Determine the Any KindOfs present to be forbidden from Salvaging Weapon Flags. Can be configured with NONE for no qualifications. Defaults to: Not Configured.
- Added RequiredKindOfArmorSalvager. Determine the Multiples KindOfs required to qualify for Salvaging Weapon Flags. Can be configured with NONE for no qualifications. Defaults to: ARMOR_SALVAGER.
- Added ForbiddenKindOfArmorSalvager. Determine the Any KindOfs present to be forbidden from Salvaging Weapon Flags. Can be configured with NONE for no qualifications. Defaults to: Not Configured.

```
{ "WeaponSetFlags", INI::parseIndexListVector, WeaponSetFlags::getBitNames(),offsetof(
{ "WeaponModelConditionFlags",  INI::parseIndexListVector,  ModelConditionFlags::getBit
{ "ArmorSetFlags", INI::parseIndexListVector,   ArmorSetFlags::getBitNames(),offsetof(
{ "ArmorModelConditionFlags",   INI::parseIndexListVector,  ModelConditionFlags::getBit
{ "GrantsMultipleBonusesOnPickup",  INI::parseBool,     NULL, offsetof( SalvageCrateCol
```

- Added WeaponSetFlags. Determines the Weapon Set Flags given. Can be customized with more than 2 flags. The first Flag determines the First Level and the last Flag determines the Max Level for Weapon Set Flags. Defaults to: CRATEUPGRADE_ONE, CRATEUPGRADE_TWO.
- Added WeaponModelConditionFlags. Determines the Model Condition Flags granted when receiving Weapon Set Flags from the Salvage Crate. The Flags granted are based on the current level of WeaponSetFlags received. Does nothing if the current Level of WeaponSetFlags received is greater than the configured Model Flags. Defaults to: WEAPONSET_CRATEUPGRADE_ONE, WEAPONSET_CRATEUPGRADE_TWO.
- Added ArmorSetFlags. Same as the above, but for Armor Sets. Defaults to: CRATE_UPGRADE_ONE, CRATE_UPGRADE_TWO.
- Added ArmorModelConditionFlags. Ditto, but for WeaponModelConditionFlags. Defaults to: ARMORSET_CRATEUPGRADE_ONE, ARMORSET_CRATEUPGRADE_TWO.
- Added GrantsMultipleBonusesOnPickup. Determine whether the Salvage Crate grants multiple bonuses instead of just one while pickup (See below.) Defaults to: No.

```
{ "PickupOrder",    INI::parseIndexListVector,  TheSalvagePickupNames, offsetof( Salvage
{ "MultiPickupFlags",   INI::parseBitString32,  TheSalvagePickupNames, offsetof( Salvage
{ "MultiPickupAmountModifierAfterCompletion",   parseAmountModifier,    NULL, offsetof(
{ "MultiPickupChanceModifierAfterCompletion",   parseChanceModifier,    NULL, offsetof(
```

- Added PickupOrder. Determine the order of Bonuses granted by the Crate. Starts from the first Bonus and Moves on to the next when the picker has the Max Level of the current Bonus. When configured, it wipes the previous order and assigns the new order. If GrantsMultipleBonusesOnPickup is configured with ‘Yes’, ignore the Pickup Order and use ‘MultiplePickupFlags’. Defaults to: ARMOR, WEAPON, LEVEL, MONEY.
- Added MultiplePickupFlags. Used when ‘GrantsMultipleBonusesOnPickup’ is configured to ‘Yes’. Determines which bonuses would be granted upon picking up the salvage crate. Defaults to: Not Configured.

- Added MultiPickupAmountModifierAfterCompletion. Used only when ‘GrantsMultipleBonusesOnPickup’ is configured to ‘Yes’. If one of the bonuses granted from ‘MultiplePickupFlags’ has reached its Max Level, then other bonuses can be configured with a boost to their amount. If the granted level goes beyond the Max Level while currently it is not Maxed Out, it will grant the Max Level. Defaults to: Not Configured.

  You are required to declare in the format of: COMPLETION [Bonus to Max Out] BONUS [Bonuses to grant Amount] [Amount to Grant].

  Multiple Declarations of Bonuses can Stack! Can be configured with Negative Values to decrease the granted amount. Can be redeclared Multiple times, each for the Completion of Different Bonuses.

  COMPLETION can be configured with anything but MONEY and OCL, which cannot reach the Max Level and are designed to cause crashes when declared for Completion.

  When Bonuses are configured with ARMOR, WEAPON, UPGRADES, SCIENCE, STATUS, WEAPON_BONUS or OCL, it will Increment the amount of Levels to Skip for the Declared Bonus when receiving it from Pickup.

  NOTE: You are also required to configure MIN and MAX for points amounts beside MONEY, SCIENCE_POINTS and SKILL_POINTS as they are configured with Min and Max Amounts to give (Configure the same value if you want to give a constant non-random value.)

  The format for declaration is shown below.

Example:

Behavior = SalvageCrateCollide ModuleTag_02

…

MultiPickupAmountModifierAfterCompletion = COMPLETION WEAPON BONUS MONEY MIN +999 MAX +9999 LEVEL +999

MultiPickupAmountModifierAfterCompletion = COMPLETION SCIENCE BONUS MONEY MIN +1999 MAX +19999 WEAPON_BONUS 2 OCL 1

…

End

The declaration states that when WeaponSetFlags has been set to the Last or Max Level, then the Amount of Money and Levels received would receive an extra +999 amount.

Then it was redeclared for SCIENCE, which means you have all the Sciences that are declared within the ‘SciencesGranted’ variables of the Module. The bonus given is then the same as declared, only skip two levels when Picking Up Crates that gives WEAPON_BONUS and one level for OCL.

- Added MultiPickupAmountChanceAfterCompletion. Used only when ‘GrantsMultipleBonusesOnPickup’ is configured to ‘Yes’. If one of the bonuses granted from ‘MultiplePickupFlags’ has reached its Max Level, then other bonuses can be configured with a boost to their pickup Chance. The format for declaration is shown below, similar to ‘MultiPickupAmountModifierAfterCompletion’. Defaults to: Not Configured.

  You are required to declare in the format of COMPLETION [Bonus to Max Out] BONUS [Bonuses to grant Chances] [Chance to Grant].

  Multiple Declarations of Bonuses can Stack! Can be configured with Negative Values to decrease the granted chance. Can be redeclared Multiple times, each for the Completion of Different Bonuses.

  COMPLETION can be configured with anything but MONEY and OCL, which cannot reach the Max Level and are designed to cause crashes when declared for Completion.

  BONUS granted cannot be configured with ARMOR, or MONEY. As ARMOR and MONEY will always have a 100% chance to be granted. Though, they can only be disabled when not declared under MultiplePickupFlags or PickupOrder.

  The format for declaration is shown below.

Example:

Behavior = SalvageCrateCollide ModuleTag_02

…

MultiPickupChanceModifierAfterCompletion = COMPLETION ARMOR LEVEL +78% OCL +12%

…

End

The declaration states that when ArmorSetFlags has been set to the Last or Max Level, then the Chances of getting Levels for Pickup will be increased by +78% and OCL by +12%.

```
{ "GrantWeaponBonus",        parseWeaponBonus, NULL, offsetof( SalvageCrateCollideM
{ "GrantStatus",             parseStatus, NULL, offsetof( SalvageCrateCollideModule
{ "GrantUpgrades",       INI::parseAsciiStringVector, NULL, offsetof( SalvageCrateC
{ "SciencesGranted", INI::parseScienceVector, NULL, offsetof( SalvageCrateCollideM
{ "OCLs",                INI::parseObjectCreationListVector, NULL,   offsetof( Salv
```

- Added GrantWeaponBonus. Determines the Weapon Bonuses given. Like WeaponSetFlags and ArmorSetFlags, Subsequent Weapon Bonuses determine the extra levels declared for the Module. The first Flag determines the First Level and the last Flag determines the Max Level. Defaults to: Not configured.
- Added GrantStatus. Determines the Statuses given. Subsequent Statuses determine the extra levels declared for the Module. The first Flag determines the First Level and the last Flag determines the Max Level. Defaults to: Not configured.
- Added GrantUpgrades. Determines the Upgrades given. Subsequent Upgrades determine the extra levels declared for the Module. The first Flag determines the First Level and the last Flag determines the Max Level. Defaults to: Not configured.
- Added SciencesGranted. Determines the Sciences given. Subsequent Statuses determine the extra levels declared for the Module. The first Flag determines the First Level and the last Flag determines the Max Level. Defaults to: Not configured.
- Added OCLs. Determines the OCLs given. Subsequent Statuses determine the extra levels declared for the Module. Contrary to the modules above, this does not use a Level system and will always spawn the first OCL in the list. It also does not have a Max Level, but can be configured with multiple levels. ‘MultiPickupAmountModifierAfterCompletion’ can add to the Levels for spawning. Defaults to: Not configured.

```
{ "RemovePreviousWeaponBonus",  INI::parseBool, NULL, offsetof( SalvageCrateCollide
{ "RemovePreviousStatus",       INI::parseBool, NULL, offsetof( SalvageCrateCollide
{ "RemovePreviousUpgrade",      INI::parseBool, NULL, offsetof( SalvageCrateCollide
```

- Added RemovePreviousWeaponBonus. Determines whether receiving newer levels of Weapon Bonus from the Module removes previous Weapon Bonus granted by the module. Does not affect the level progression in any way. Defaults to: No.
- Added RemovePreviousStatus. Determines whether receiving newer levels of Status from the Module removes previous Status granted by the module. Does not affect the level progression in any way. Defaults to: No.

- Added RemovePreviousWeaponUpgrade. Determines whether receiving newer levels of Upgrades from the Module removes previous Upgrades granted by the module. Does not affect the level progression in any way. Defaults to: No.

```
{ "MinimumSciencePurchasePointsGranted", INI::parseInt, NULL, offsetof( SalvageCrate
{ "MaximumSciencePurchasePointsGranted", INI::parseInt, NULL, offsetof( SalvageCrate
{ "MinimumSkillPointsGranted", INI::parseInt, NULL, offsetof( SalvageCrateCollideMod
{ "MaximumSkillPointsGranted", INI::parseInt, NULL, offsetof( SalvageCrateCollideMod
```

- Added MinimumSciencePurchasePointsGranted. Determines the minimum Science Purchase Points to be granted by the module. It reaches 'Max' when the Player has enough Science Purchase Points to purchase all General's Powers. Defaults to: 0 or not configured.
- Added MAXimumSciencePurchasePointsGranted. Determines the maximum Science Purchase Points to be granted by the module. It reaches 'Max' when the Player has enough Science Purchase Points to purchase all General's Powers. Defaults to: 0 or not configured.
- Added MinimumSkillPointsGranted. Determines the minimum Skill Points to be granted by the module. It reaches 'Max' when the Player has reached the Final Rank configured by the game. Defaults to: 0 or not configured.
- Added MaximumSkillPointsGranted. Determines the maximum Skill Points to be granted by the module. It reaches 'Max' when the Player has reached the Final Rank configured by the game. Defaults to: 0 or not configured.

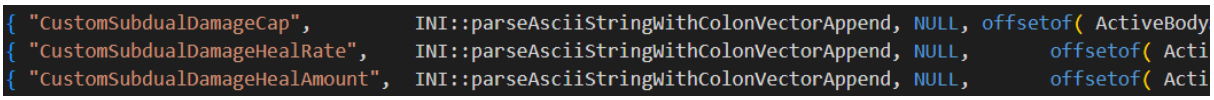
```
{ "WeaponBonusChance",  INI::parsePercentToReal,    NULL, offsetof( SalvageCrate
{ "StatusChance",   INI::parsePercentToReal,    NULL, offsetof( SalvageCrateColl
{ "UpgradeChance",  INI::parsePercentToReal,    NULL, offsetof( SalvageCrateColl
{ "ScienceChance",  INI::parsePercentToReal,    NULL, offsetof( SalvageCrateColl
{ "SciencePurchasePointsChance",    INI::parsePercentToReal,    NULL, offsetof(
{ "SkillPointsChance",  INI::parsePercentToReal,    NULL, offsetof( SalvageCrate
{ "OCLChance",  INI::parsePercentToReal,    NULL, offsetof( SalvageCrateCollideM
```

- Added WeaponBonusChance. Determines the chance to receive a Weapon Bonus when picking up the Crate. Defaults to: 100%.
- Added StatusChance. Determines the chance to receive a Status when picking up the Crate. Defaults to: 100%.
- Added UpgradeChance. Determines the chance to receive an Upgrade when picking up the Crate. Defaults to: 100%
- Added ScienceChance. Determines the chance to receive a Science when picking up the Crate. Defaults to: 100%.

- Added SciencePurchasePointsChance. Determines the chance to receive SciencePurchasePoints when picking up the Crate. Defaults to: 100%.
- Added SkillPointsChance. Determines the chance to receive Skill Points when picking up the Crate. Defaults to: 100%.
- Added OCLChance. Determines the chance to spawn an OCL when picking up the Crate. Defaults to: 100%.

BodyModules:

- Active Body:



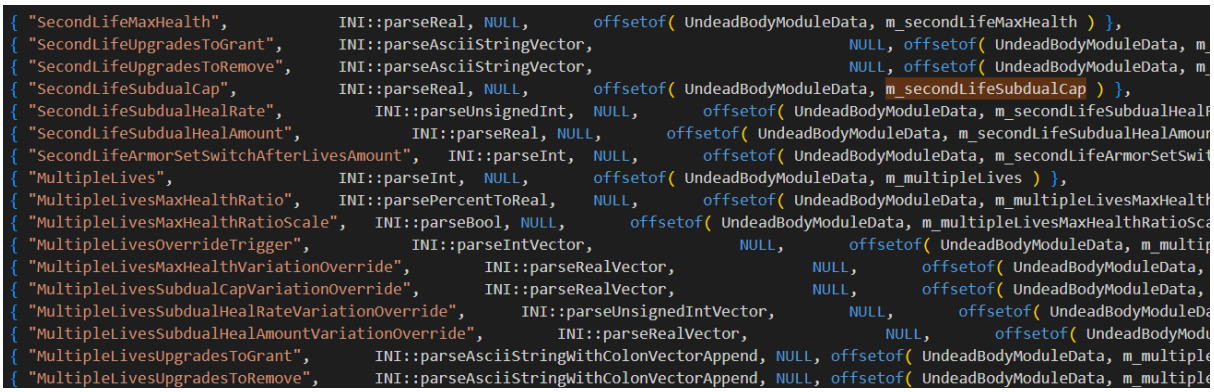
- Added CustomSubdualDamageCap, CustomSubdualDamageHealRate, and CustomSubdualDamageHeal for interacting with Custom Subdual Types. Configure in the same format as their normal Subdual counterparts and react respectively to the Custom Subdual Types configured. If not configured, Custom Subdual Types will use SubdualDamage amount for calculation.
- Declared in array format, for example:

CustomSubdualDamageCap = RedGuardEMP:1000 CUSTOMECM:10000 NewDisable:8235

CustomSubdualDamageHealRate = CUSTOMECM:10

CustomSubdualDamageHealAmount = NewCustomEMP:1000

- Undead Body:



- ~~Changes: Changes to more accommodate Infantry to use UndeadBody. PoisonedBehavior that deals UNRESISTABLE DamageTypes can be configured to not Remove Second Life Trigger.~~

  Bug Fixed by hanfield.
- More memory pool has been allocated to UndeadBody; {32,32} to {512, 128}
- Added SecondLifeUpgradesToGrant, and SecondLifeUpgradesToRemove:

  You can now customize the Upgrades Granted or To Remove from the unit when triggering Second Life.

- Added SecondLifeSubdualCap, SecondLifeSubdualHealRate, and SecondLifeSubdualHealAmount:

  You can now customize the Subdual Damage Cap, Heal Rate and Heal Amount after triggering Second Life.

  Note: Only the Subdual Damage Cap can be set as ‘0’. Setting the value to ‘0’ will make the Object Immune to Subdual Damage. Others are required to be set to more than 0 to register.

- Added SecondLifeArmorSetSwitchAfterLivesAmount:

  You can customize the amount of times for Multiple Lives required to trigger for setting the Armor Set to SECOND_LIFE. (INT value. Defaults to 0, which triggers instantly after triggering Second Life.)

- Added MultipleLives:

  You can now add Multiple Lives to the Unit. Multiple Lives account for how many more times the object can trigger Undead Body after Triggering Second Life. (Example: Setting MultipleLives = 3 enables the unit to be revived 3 more times after its Second Life. This gives the unit a total of 5 lives; Default + Second Life + 3 Multiple Lives.)

  Note: For Units to function properly while triggering Multiple Lives, the ‘SleepsAfterAmountofDeaths’ and ‘DeathOnNoPassengersAmountOfDeaths’ under BattleBusSlowDeathBehavior must also be configured.

- Added MultipleLivesMaxHealthRatio, and MultipleLivesMaxHealthRatioScale:

  You can now customize the Ratio of the Max Health per Revive when the unit triggers Multiple Lives. Setting MultipleLivesMaxHealthRatio according to a certain coefficient (%) means it will account for the % of health determined by SecondLifeMaxHealth. This can be over 100% to buff the Unit’s Max Health if it dies after the Second Life.

  If MultipleLivesMaxHealthRatioScale is set to ‘Yes’, then the percentage of MultipleLivesMaxHealthRatio is scaled according to the number of times Multiple Lives is triggered. (Defaults to ‘No’.)

- Added MultipleLivesOverrideTrigger, MultipleLivesMaxHealthVariationOverride, MultipleLivesSubdualCapVariationOverride, MultipleLivesSubdualHealRateVariationOverride, and MultipleLivesSubdualHealAmountVariationOverride:

  Additionally, you can also Override the Max Health given at a specific value of Multiple Lives with MultipleLivesOverrideTrigger. When MultipleLivesOverrideTrigger is set to a value, and its given

MultipleLivesMaxHealthVariationOverride is higher than 0, it will override the Max Health of the Unit after triggering Multiple Lives. The same can be applied to SubdualCap, SubdualHealRate, and SubdualHealAmount. Setting the number to 0 does not override the value at the given moment.

- Added MultipleLivesUpgradesToGrant, and MultipleLivesUpgradesToRemove:

  Functions exactly like Added SecondLifeUpgradesToGrant, and SecondLifeUpgradesToRemove, but for Multiple Lives. The registration format for this variable is slightly different. (See below.)

Example for MultipleLives feature,

- If the declared value is MultipleLivesOverrideTrigger = 9 3 7 6 1. If your Death Amount is 6. You would Trigger the Condition at '6', despite 9 being assigned the first value.
- If the declared value is MultipleLivesMaxHealthVariationOverride = 200 500 0 100 .
- At 1st time triggering MultipleLives, it will use the SecondLifeMaxHealth value because the value is not declared. At 3rd death, the Max Health will be set as ‘500’. At 6th death, the Max Health will be set as ‘100’. At 7th death, since the value is ‘0’, it does not register, thus Max Health will be set as the value configured for SecondLifeMaxHealth. At the 9th death, the Max Health will be set as ‘200’. Any other times MultipleLives triggered that are not declared will use SecondLifeMaxHealth amount as its value.
- You can also add more Upgrades to Grant or Remove at specific time you Trigger Multiple Lives. The parsing value can be used as example below:

MultipleLivesUpgradesToGrant = 1:Upgrade1 Upgrade2 2 Upgrade64 Upgrade66 3: Upgrade645

MultipleLivesUpgradesToGrant = 4:Upgrade33 Upgrade24

This means when triggering the 1st Multiple Life, Upgrade1 & Upgrade2 would be granted.

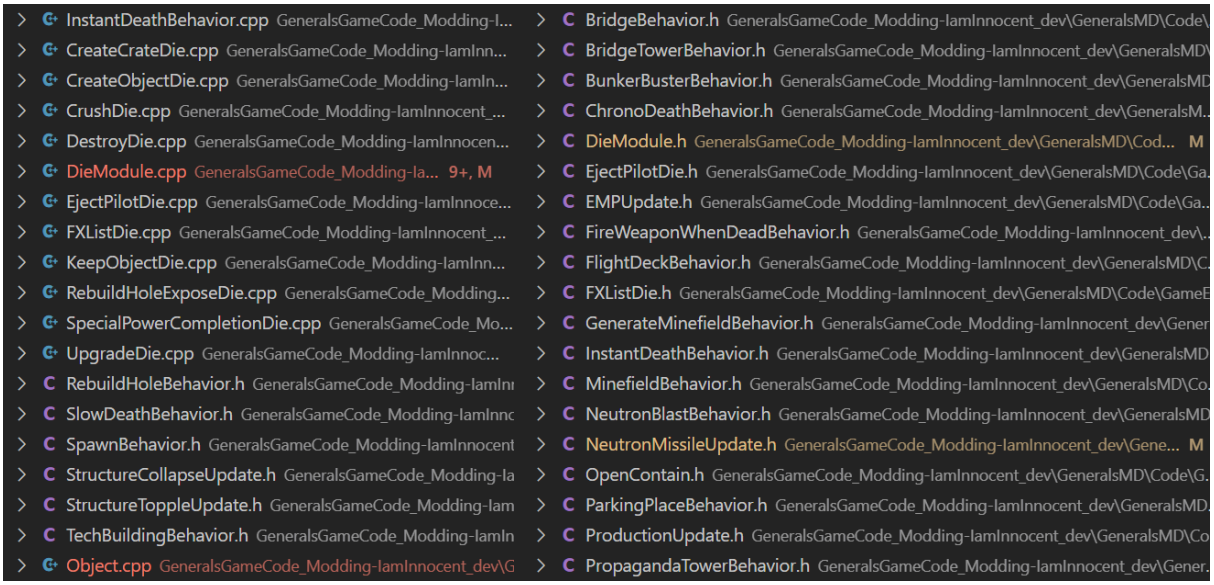
At the 2nd Multiple Life, Upgrade64, and Upgrade66 would be granted.

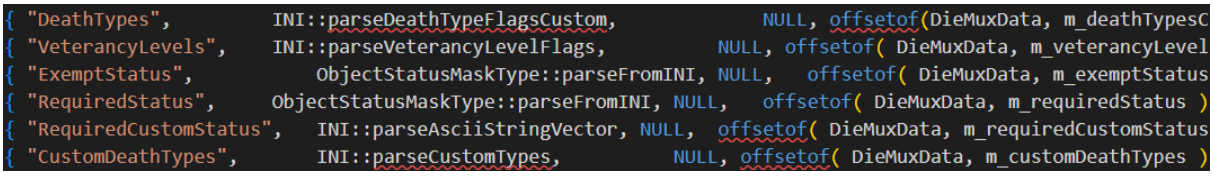
At the 3rd, Upgrade645 would be granted.

At the 4th, Upgrade33, and Upgrade24 would be granted.

DieModules:

- List of Modules that uses DieModules:





- Added RequiredCustomStatus. Enable Custom Statuses to work with DieModules. Functions the same way as the existing RequiredStatus function.
- Various functions use DieModules, but the features above are more commonly used with SlowDeathBehavior, InstantDeathBehavior and DestroyDie.

- BattleBusSlowDeathBehavior:

- More memory pool has been allocated to BattleBusSlowDeathBehavior; {64,32} to {256, 32}
- Added PercentDamageToPassengersScale, and PercentDamageToPassengersScaleRatio:

  You can now customize whether PercentDamageToPassengers scales according to the number of deaths triggered, by setting it to 'Yes'. However, you must configure the Ratio to scale first in (%). Lower Ratio indicates the damage diminishes with each death. A higher ratio indicates the damage increases with each death.

- Added SleepsAfterAmountofDeaths, ModelConditionSwitchAfterAmountofDeaths TriggerFXAfterAmountofDeaths, TriggerOCLAfterAmountofDeaths, ThrowForceAfterAmountofDeaths, and PercentDamageToPassengersAfterAmountofDeaths:

  Each of the attributes such as Sleep/Disabled, Model Condition Switch, FX, OCL, throwForce and PercentDamageToPassengers can be configured to trigger after a certain amount of deaths. (Default: 0, but when triggering Second Life, the amount of deaths altogether is 1, any trigger of Multiple Lives will add 1 to the current amounts of deaths.) Example: A unit which has triggered Multiple Lives 3 times and is still alive has a total of 4 Amount of Deaths. 0 while not triggering Second Life, 1 after triggering Second Life, 2 after triggering the 1st Multiple Life, and so on.

  Note: It is recommended to put Sleep/Disable and Model Condition Switch with same amount of deaths, since Battle Bus's SECOND_LIFE model does not have moving animations

- Added FXStartUndeathAmountDeathsOverride, OCLStartUndeathAmountDeathsOverride, FXHitGroundAmountDeathsOverride, and OCLHitGroundAmountDeathsOverride:
  You can now override the FX and OCL when reaching certain amounts of deaths. They can be declared multiple times like UpgradeOCL.

Note: When a Unit Sleeps, it no longer triggers FXHitGround and OCLHitGround.

For example:

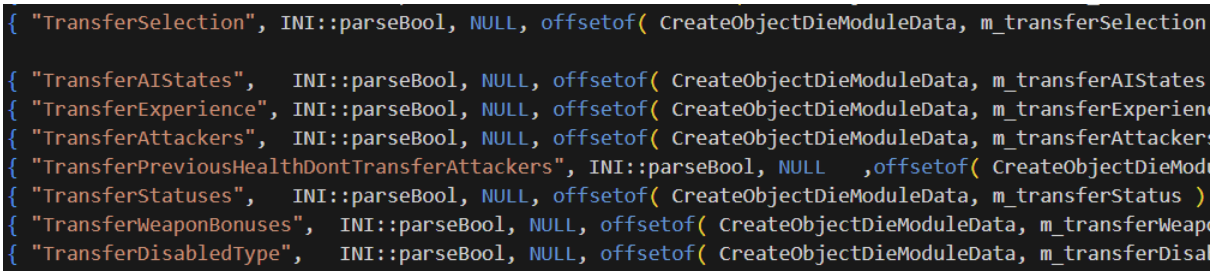
FXHitGroundAmountDeathsOverride = 3 NewFX

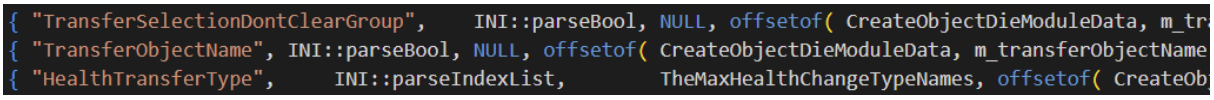
Causes the Unit to spawn NewFX when Hitting the Ground from the air after reaching death no.3, assuming the SleepsAfterAmountofDeaths is larger than 3.

- Added DeathOnNoPassengersAmountOfDeaths, which is the expanded default death check mechanic for BattleBusSlowDeathBehavior. It now prevents the Object from triggering Second Life if there are no Passengers Contained within the Unit. Configured with AmountofDeaths required for this feature to Trigger. Defaults to ‘2’, which Triggers after the unit is destroyed after triggering Second Life.
- The mechanic overlaps with ‘EmptyHulkDestructionDelay’, which determines the death timer after triggering Second Life if there are no passengers within the Unit.

- CreateObjectDie:

- Added various configuring Variables for Objects Spawned.





- TransferSelection (Added By TheSuperHackers) is adjusted to check if the Object is Selected rather than it requiring to be the First Selected.
- Added "TransferAIStates". Does the Object transfer its AI States (Moving, Attacking, Constructing, Repairing) unto the Object Created? Defaults to: No.
- Added "TransferExperience". Does the Object transfer its Experience Points unto the Object Created? Defaults to: No.
- Added "TransferAttackers". Does the Module tell recent Attackers of the Object to attack the Object Created? Defaults to: No.
- Added "TransferPreviousHealthDontTransferAttackeers". When ‘TransferHealth’ is configured, it also by default Transfer the Attackers onto the Object Created, regardless if ‘TransferAttackers’ is Set to ‘No’. Setting this to ‘Yes’ enables the module to Not Transfer Attackers when ‘‘TransferAttackers’ ‘ is Set to ‘No’. Defaults to: No.
- Added "TransferStatuses". Does the Object transfer its Statuses unto the Object Created? Defaults to: No.

- Added "TransferWeaponBonuses". Does the Object transfer its Weapon Bonuses unto the Object Created? Defaults to: No.
- Added "TransferDisabledType". Does the Object transfer its DisabledTypes unto the Object Created? Defaults to: No.
- Added "TransferSelectionDontClearGroup". Does 'TransferSelection', when Triggered, clear the current Selection Group? Defaults to: No.
- Added "TransferObjectName". Does the Object transfer ObjectName unto the Object Created? Enabling Scripts to identify the Created Object as the current Object. Defaults to: No.
- Added "HealthTransferType". Determines the Health Inherit Type for "TransferHealth" of the Module. Different Configuration Types:
  - PRESERVE_RATIO : Changes the Health according to the Health Ratio.
  - SAME_CURRENTHEALTH : Changes the Health to Match the Object's Current Health.
  - ADD_CURRENT_DAMAGE : Changes the Missing Health to the Object's Current Missing Health (Can kill if Missing Health exceeds Max Health.)
  - ADD_CURRENT_DAMAGE_NON_LETHAL: Changes the Missing Health to the Object's Current Missing Health (If Missing Health exceeds Max Health, Object's Health is set to 1.)
  - Defaults to: ADD_CURRENT_DAMAGE.

```
{ "TransferBombs",  INI::parseBool, NULL, offsetof( CreateObjectDieModuleData, m_transferBombs )
{ "TransferHijackers",  INI::parseBool, NULL, offsetof( CreateObjectDieModuleData, m_transferHij
{ "TransferEquippers",  INI::parseBool, NULL, offsetof( CreateObjectDieModuleData, m_transferEqu
{ "TransferParasites",  INI::parseBool, NULL, offsetof( CreateObjectDieModuleData, m_transferPar
{ "TransferPassengers", INI::parseBool, NULL, offsetof( CreateObjectDieModuleData, m_transferPas
{ "TransferToAssaultTransport", INI::parseBool, NULL, offsetof( CreateObjectDieModuleData, m_tra
{ "TransferShieldedTargets",    INI::parseBool, NULL, offsetof( CreateObjectDieModuleData, m_tra
{ "TransferShieldingTargets",   INI::parseBool, NULL, offsetof( CreateObjectDieModuleData, m_tra
```

- Added "TransferBombs". If Set to 'Yes', transfer any Sticky Bombs from the Object that Triggers the Module unto the Spawned Object. Defaults to: No.
- Added "TransferHijackers". If Set to 'Yes', transfer any Hijackers from the Object that Triggers the Module unto the Spawned Object. Defaults to: Yes.
- Added "TransferEquippers". If Set to 'Yes', transfer any Equippers, excluding Parasites, from the Object that Triggers the Module unto the Spawned Object. Defaults to: Yes.

- Added "TransferParasites". If Set to ‘Yes’, transfer any Parasites from the Object that Triggers the Module unto the Spawned Object. Defaults to: Yes.
- Added "TransferPassengers". If Set to ‘Yes’, transfer any Passengers from the Object that Triggers the Module unto the Spawned Object. If not, the Passengers will be Ejected when the Object has Died. Defaults to: Yes.
- Added "TransferToAssaultTransport". If Set to ‘Yes’, transfer the Object’s Assault Transport status onto the Spawned Object and Remove it from Assault Transport. Defaults to: No.
- Added "TransferShieldedTargets". If Set to ‘Yes’, transfer the Spawner’s SHIELDED_TARGET Status to the Spawned Objects along with the Shielder’s information and ProtectionTypes. Status and Information is then removed from the Spawner. Defaults to: No.
- Added "TransferShieldingTargets". If Set to ‘Yes’, transfer the Spawner’s Current Shielding Target’s Information to the Spawned Objects. Shielding Information is then removed from the Spawner. Defaults to: No.

```
{ "OrientInForceDirection", INI::parseBool, NULL, offsetof(CreateObjectDieModuleData, m_orien
{ "ExtraBounciness",              INI::parseReal,                    NULL, offsetof( Creat
{ "ExtraFriction",            parseFrictionPerSec,                       NULL, offsetof( C
{ "Offset",                   INI::parseCoord3D,             NULL, offsetof( CreateObjectD
{ "Disposition",           INI::parseBitString32,         DispositionNames, offsetof( Creat
{ "DispositionIntensity",  INI::parseReal,                    NULL,   offsetof( CreateObjec
{ "SpinRate",                 INI::parseAngularVelocityReal,  NULL, offsetof(CreateObjectDi
{ "YawRate",                  INI::parseAngularVelocityReal,  NULL, offsetof(CreateObjectDi
{ "RollRate",                 INI::parseAngularVelocityReal,  NULL, offsetof(CreateObjectDi
{ "PitchRate",             INI::parseAngularVelocityReal,  NULL, offsetof(CreateObjectDieMod
{ "MinForceMagnitude",  INI::parseReal, NULL, offsetof(CreateObjectDieModuleData, m_minMag) }
{ "MaxForceMagnitude",  INI::parseReal, NULL, offsetof(CreateObjectDieModuleData, m_maxMag) }
{ "MinForcePitch",  INI::parseAngleReal,    NULL, offsetof(CreateObjectDieModuleData, m_minPi
{ "MaxForcePitch",  INI::parseAngleReal,    NULL, offsetof(CreateObjectDieModuleData, m_maxPi
{ "DiesOnBadLand",  INI::parseBool, NULL, offsetof(CreateObjectDieModuleData, m_diesOnBadLand
```

- Added ObjectCreationList’s Disposition and Spawning Variables onto CreateObjectDie Module.

UpdateModules:

General:

- Several UpdateModules have been reworked to be Sleepy. Previously All UpdateModules would run every frame with/without commands, some are also not optimized and would be terrible for performance.
- These Update modules have been optimized to be Sleepy:
  - AutoDepositUpdate
  - AutoFindHealingUpdate
  - BoneFXUpdate
  - CheckpointUpdate
  - CleanupHazardUpdate
  - DemoTrapUpdate
  - DynamicGeometryUpdate
  - EMPUpdate
  - EnemyNearUpdate
  - FloatUpdate
  - HelicopterSlowDeathBehavior
  - HijackerUpdate
  - JetSlowDeathBehavior
  - MobMemberSlavedUpdate
  - NeutronMissileSlowDeathBehavior
  - OCLUpdate
  - PointDefenseLaserUpdate
  - ProductionUpdate
  - PropagandaTowerBehavior
  - ProneUpdate
  - RadarUpdate
  - SlavedUpdate
  - SpawnBehavior
  - StealthUpdate
  - WaveGuideUpdate

- CountermeasuresBehavior:

- Edited Countermeasures to also work with DumbProjectileBehavior, FreeFallProjectileBehavior, and NeutronMissileUpdate. If they are configured with SMALL_MISSILE, Countermeasures will destroy them after reaching the MissileDecoyDelay frame limit. Destroying Projectiles with DumbProjectileBehavior, or FreeFallProjectileBehavior will cause them to deal no

damage, but Projectiles with NeutronMissileUpdate can still deal damage, as the damage is stored in their SlowDeathBehavior.

- Jets no longer reload instantly on landing on their respective airfields. They now respect ‘ReloadTime’ for Countermeasures. The reloading mechanic for the default behavior is preserved. As long as the ‘ReloadTime’ is 0 or not configured, units will always reload instantly when fully landed.

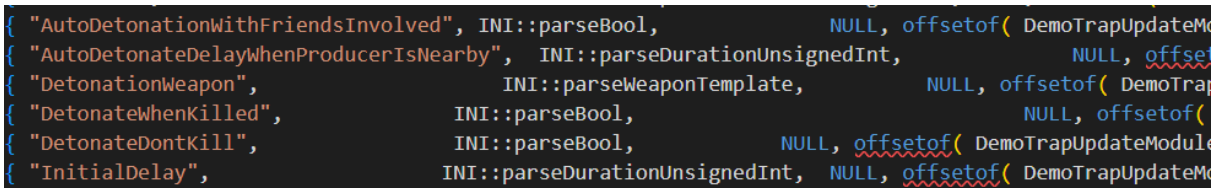
```
{ "MustReloadAtAirfield",       INI::parseBool,                              NULL, offsetof( CountermeasuresBehavi
{ "MissileDecoyDelay",          INI::parseDurationUnsignedInt,  NULL, offsetof( CountermeasuresBehaviorModuleData, m_
{ "ReactionLaunchLatency",  INI::parseDurationUnsignedInt,  NULL, offsetof( CountermeasuresBehaviorModuleData, m_coun

{ "StartsActive",   INI::parseBool,               NULL, offsetof( CountermeasuresBehaviorModuleData, m_initiallyAct
{ "VolleyLimitPerMissile",  INI::parseInt,                  NULL, offsetof( CountermeasuresBehaviorModuleData, m_voll
{ "ContinuousVolleyInAir",  INI::parseBool,                 NULL, offsetof( CountermeasuresBehaviorModuleData, m_cont
{ "ReactingToKindOfs",  KindOfMaskType::parseFromINI,   NULL, offsetof( CountermeasuresBehaviorModuleData, m_reacting
{ "NoAirboneCountermeasures",   INI::parseBool,                NULL, offsetof( CountermeasuresBehaviorModuleData, m_
{ "GroundCountermeasures",  INI::parseBool,                 NULL, offsetof( CountermeasuresBehaviorModuleData, m_cons
{ "NonTrackingDetonateDistance", INI::parseUnsignedInt,                NULL, offsetof( CountermeasuresBehaviorModule
{ "MustReloadAtBarracks",       INI::parseBool,                              NULL, offsetof( CountermeasuresBehavi
{ "MustReloadNearRepairDocks",      INI::parseBool,                          NULL, offsetof( CountermeasuresBehavi
{ "MustReloadObjectDistance",   INI::parseReal,                         NULL,       offsetof( CountermeasuresBehavior
{ "MustReloadNearObjects",  INI::parseAsciiStringVector,                NULL,       offsetof( CountermeasuresBehavior
```

- Restored MustReloadAtAirfield function. If set to ‘Yes’, Countermeasures can only be reloaded at the Airfield if ‘ReloadTime’ is set more than 0. The feature is not required to be re-declared on every unit because by default, Countermeasures won’t reload automatically if ‘ReloadTime’ is set to 0 or not configured.
- Added StartsActive, when set to ‘Yes’, enables the Object to use Countermeasures without any Upgrades Required.
- Added VolleyLimitPerMissile. Configure the amount of volleys or flares the Object is able to launch to divert a missile. Defaults to ‘0’, or unlimited.
  NOTE: Setting at unlimited while having faster ‘ReloadTime’ than the total Volleys for ‘DelayBetweenVolleys’ will cause the Object to Continuously fire flares despite if the Object is not being aimed at. It’s best to configure the value at 4 if you plan to do unlimited Countermeasures for a unit.
  To disable the Object to launch volleys or flares, declare ‘-1’ to the value.
- Added ContinuousVolleyInAir, when set to ‘Yes’, continues to launch volleys or flares after targeted with the redirectable projectiles while the Object is in Air. Defaults to ‘Yes’. This is a default mechanic that is adjusted to be modifiable.
- Added ReactingToKindOfs, which enables Countermeasures to react to any projectiles with the KindOfs declared. This is compatible with the projectile behaviors reworked above. Defaults to: NONE. Reacts to SMALL_MISSILE if not configured.

- Added NoAirborneCountermeasures. Disables the Object from using Countermeasures if it is airborne. Defaults to: No.
- Added GroundCountermeasures. Enables the Object to use Countermeasures if it is on ground. Defaults to: No.
- Added NonTrackingDetonateDistance. Set a distance for Non Tracking Projectiles, such as DumbProjectileBehavior to detonate when reaching near the target instead of instantly when Countermeasure triggers. Defaults to '0', which triggers instantly after Countermeasure triggers.
  Note: After testing, the optimal trigger distance to use is 25-30.
- NOTE: Non Tracking Projectiles triggering Countermeasures still deal damage unless detonated with this mechanic.
- Added MustReloadAtBarracks. Like MustReloadAtAirfield, but for Barracks. Requires Infantry to go into the Barracks to Heal to reload Countermeasures. Reloads Instantly after Exit. Defaults to: No.
- Added MustReloadNearDocks. Like MustReloadAtAirfield, but requires the Object to be near a Repair Dock (War Factory) to reload Countermeasures. Defaults to: No.
- Added MustReloadObjectDistance. Sets the range for MustReloadNearDocks or MustReloadNearObjects to trigger the Object's Countermeasure reload time. Defaults to: 100 units.
- Added MustReloadNearObjects. Like MustReloadNearDocks, but requires the Object to be near any one of the declared Objects reload Countermeasures. Declares Objects for this variable instead of 'Yes' or 'No'

- DemoTrapUpdate:



- Added AutoDetonateDelayWhenProducerIsNearby. If AutoDetonationWithFriendsInvoled is set to 'No', the Demo Trap won't detonate when an Ally is nearby even though an enemy is within the Trigger range. This applies to the Builder or Producer of the Demo Trap, but only during the configured duration. Defaults to: 0 or triggers instantly.

- Added DetonateDontKill. The module detonates itself after an enemy is within the Trigger range. This variable enables the Object to only fire its DetonationWeapon (if configured) and not detonate itself. Defaults to: No.
- Added InitialDelay. Sets the delay for this module to be active after being Created or Builded. Defaults to: 0 or no delay.

- FireWeaponUpdate:

- Now gets the Container or Converted/Hijacked/Equipped Object's Position when considering its Position to fire.
- Now respects the Weapon Template's Upgrade and Status requirements when checking whether it should fire.

- FireWeaponAdvancedUpdate (Modding Devs' Implemented Feature):

- Now gets the Container or Converted/Hijacked/Equipped Object's Position when considering its Position to fire.
- Now respects the Weapon Template's Upgrade and Status requirements when checking whether it should fire.

- FireWeaponWhenDamagedBehavior:

- Memory Pool increased from {32,32} to {256, 64}.
- Now gets the Container or Converted/Hijacked/Equipped Object's Position when considering its Position to fire.

```
{ "CustomDamageTypes", INI::parseCustomTypes, NULL, offsetof( FireWeaponWhenDamagedBehaviorModuleData, m_cu
{ "RequiredStatus",     ObjectStatusMaskType::parseFromINI, NULL, offsetof( FireWeaponWhenDamagedBehaviorMo
{ "ForbiddenStatus",    ObjectStatusMaskType::parseFromINI, NULL, offsetof( FireWeaponWhenDamagedBehaviorMo
{ "RequiredCustomStatus",   INI::parseAsciiStringVector, NULL,  offsetof( FireWeaponWhenDamagedBehaviorModu
{ "ForbiddenCustomStatus",  INI::parseAsciiStringVector, NULL,  offsetof( FireWeaponWhenDamagedBehaviorModu
{ "WeaponSlot",     INI::parseQuotedAsciiString,    NULL, offsetof( FireWeaponWhenDamagedBehaviorModuleData
```

- Memory Pool increased from {32,32} to {256, 64}.
- Added Support for CustomDamageTypes, functions exactly like DamageTypes required for trigger but for CustomDamageTypes.
- When a unit with FireWeaponWhenDamagedBehavior is attacked with a weapon that uses a CustomDamageType, it will consider the CustomDamageTypes list, and will not consider the weapon's DamageType for Trigger.
- If a weapon does not have a declared CustomDamageType, it will use the behavior's DamageType list for trigger.

- CustomDamageTypes cannot be assigned “ALL” or “NONE”. They are required to be assigned at DamageTypes, and they cannot be reassigned at CustomDamageTypes. +CUSTOMDAMAGE1 -CUSTOMDAMAGE2 will work the same way as DamageTypes to CustomDamageTypes.
- Added WeaponSlot. You can now customize the WeaponSlot used for the mechanic, it’s used for cosmetic purposes to correct the WeaponBone and the Firing FX.
- Added RequiredStatus, ForbiddenStatus, RequiredCustomStatus, and ForbiddenCustomStatus. If declared, the Object requires to satisfy the Statuses to trigger the Behavior. Declared in the same way as SlowDeathBehavior or FireWeaponCollide.
- Note: The RequiredStatus, and RequiredCustomStatus requires having All of the declared statuses instead of any of the statuses, while ForbiddenStatus, and ForbiddenCustomStatus requires having Any of the declared statuses.

```
{ "ContinuousDurationPristine", INI::parseDurationUnsignedInt, NULL, offsetof(FireWeaponWhenDamagedBehavi
{ "ContinuousDurationDamaged", INI::parseDurationUnsignedInt, NULL, offsetof(FireWeaponWhenDamagedBehavio
{ "ContinuousDurationReallyDamaged", INI::parseDurationUnsignedInt, NULL, offsetof(FireWeaponWhenDamagedB
{ "ContinuousDurationRubble", INI::parseDurationUnsignedInt, NULL, offsetof(FireWeaponWhenDamagedBehavior
{ "ContinuousIntervalPristine", INI::parseDurationUnsignedInt, NULL, offsetof(FireWeaponWhenDamagedBehavi
{ "ContinuousIntervalDamaged", INI::parseDurationUnsignedInt, NULL, offsetof(FireWeaponWhenDamagedBehavio
{ "ContinuousIntervalReallyDamaged", INI::parseDurationUnsignedInt, NULL, offsetof(FireWeaponWhenDamagedB
{ "ContinuousIntervalRubble", INI::parseDurationUnsignedInt, NULL, offsetof(FireWeaponWhenDamagedBehavior
```

- Added ContinuousDurationPristine, ContinuousDurationDamaged, ContinuousDurationReallyDamaged, and ContinuousDurationRubble.
- You can now customize the duration for the ContinuousWeapon of FireWeaponWhenDamagedBehavior. If set to more than 1, the Module will continuously fire the Weapon according to the Object Condition until the duration runs out. Setting to ‘0’ or on Default will make the duration infinite on the Object Condition.
- Added ContinuousIntervalPristine, ContinuousIntervalDamaged, ContinuousIntervalReallyDamaged, and ContinuousIntervalRubble.
- You can now customize the interval between each time for each of the Object Conditions to fire their ContinuousWeapon.

  NOTE: You will still have to edit the weapon’s ‘DelayBetweenShots’ to work properly with each interval.

- HijackerUpdate:

> Other than Disabling SquishCollide applying to the Collider when this module is Configured, HijackerUpdate is now modified to be compatible with two modules below:

- ConvertToCarBombCrateCollide
- EquipCrateCollide

> ConvertToCarBombCrateCollide has been modified to be able to work together with HijackerUpdate. If HijackerUpdate is declared alongside, the Object is not destroyed but is “armed” into the Object, able to be removed for the CarBomb Properties to be reverted. FireWeaponUpdate and FireWeaponWhenDamagedBehavior can work on the Converted Object’s Position for ConvertToCarBombCrateCollide if HijackerUpdate module is declared alongside with it.

> ConvertToHijackedVehicleCrateCollide has been modified for the Hijacker to be able to be removed or destroyed from the Hijacked Vehicle. Removing a Hijacker from the Hijacked Vehicle treats the Vehicle as Disabled but able to be mannered, similar to Jarmen Kell’s Sniper Vehicle or Neutron Blasts.

> EquipCrateCollide, like ConvertToHijackedVehicleCrateCollide, requires HijackerUpdate for the Object to be able to Eject when the Equipped Unit has been destroyed. Unless it is configured with IsContain, if it is not configured with HijackerUpdate, the Equipper would be vulnerable with SquishCollide and the properties of EquipCrateCollide cannot be removed by reverting with StatusToRemove and StatusToDestroy.

- OverchargeBehavior:

- Enables HealthPercentToDrainPerSecond to be configured with 0% and not attempting any damage to the Unit..

```
{ "OverchargeDamageTypeFX",     DamageTypeFlags::parseSingleBitFromINI, NULL, offsetof(Overc
{ "StatusToSet",           ObjectStatusMaskType::parseFromINI, NULL, offsetof( OverchargeBe
{ "CustomStatusToSet",  INI::parseAsciiStringVector, NULL, offsetof( OverchargeBehaviorModul
{ "WeaponBonusToSet",   INI::parseWeaponBonusVector, NULL, offsetof( OverchargeBehaviorModul
{ "CustomWeaponBonusToSet",         INI::parseAsciiStringVector, NULL, offsetof( OverchargeB
{ "TintStatusToSet",           TintStatusFlags::parseSingleBitFromINI,     NULL, offsetof(
{ "CustomTintStatusToSet",  INI::parseAsciiString,  NULL, offsetof( OverchargeBehaviorModule
{ "ModelConditionToSet", ModelConditionFlags::parseFromINI, NULL, offsetof( OverchargeBehavi
```

- Added OverchargeDamageTypeFX. Enable the DamageFX of the Module to be Overwritten. The DamageType dealt is still DAMAGE_PENALTY. Defaults to: DAMAGE_PENALTY.

- Added StatusToSet. Sets the Statuses for the Object with the declared variables while Overcharge is On. Clear the Statuses when Overcharge is Off. Defaults to: None.
- Added CustomStatusToSet. Sets the Custom Statuses for the Object with the declared variables while Overcharge is On. Clear the Custom Statuses when Overcharge is Off. Defaults to: None.
- Added WeaponBonusToSet. Sets the Weapon Bonuses for the Object with the declared variables while Overcharge is On. Clear the Weapon Bonuses when Overcharge is Off. Defaults to: None.
- Added CustomWeaponBonusToSet. Sets the Custom Weapon Bonuses for the Object with the declared variables while Overcharge is On. Clear the Custom Weapon Bonuses when Overcharge is Off. Defaults to: None.
- Added TintStatusToSet. Sets the Tint Status for the Object with the declared variables while Overcharge is On. Clear the Tint Status when Overcharge is Off. Defaults to: None.
- Added CustomTintStatusToSet. Sets the Custom Tint Status for the Object with the declared variables while Overcharge is On. Clear the Custom Tint Status when Overcharge is Off. Defaults to: None.
- Added ModelConditionToSet. Sets the Model Condition for the Object with the declared variables while Overcharge is On. Clear the Model Condition when Overcharge is Off. Defaults to: None.

```
{ "ShowDescriptionLabel",    INI::parseBool,      NULL, offsetof( OverchargeBehaviorModule
{ "OverchargeOnDescriptionLabel",    INI::parseAsciiString,  NULL, offsetof( OverchargeBe
{ "OverchargeOffDescriptionLabel",   INI::parseAsciiString,  NULL, offsetof( OverchargeBe

{ "ShowOverchargeExhausted",     INI::parseBool,      NULL, offsetof( OverchargeBehaviorMo
{ "OverchargeExhaustedMessage", INI::parseAsciiString,  NULL, offsetof( OverchargeBehavi
```

- Added ShowDescriptionLabel. De-hardcoded the Overcharge Tooltip to be able to be hidden. When Set to ‘No’, hides the Overcharge Tooltip from Showing. Defaults to: Yes.
- Added OverchargeOnDescriptionLabel. Configures the Tooltip to be shown while Overcharge is On. Defaults to: ‘TOOLTIP:TooltipNukeReactorOverChargeIsOn’.
- Added OverchargeOffDescriptionLabel. Configures the Tooltip to be shown while Overcharge is Off. Defaults to: ‘TOOLTIP:TooltipNukeReactorOverChargeIsOff’.
- Added ShowOverchargeExhausted. De-hardcoded the Overcharge Unit from being Highlighted when Overcharge is Automatically turned off from ‘NotAllowedWhenHealthBelowPercent’ Feature. When Set to ‘No’, disables the

Unit from being Highlighted and the 'OverchargeExhaustedMessage' from showing on Screen when Overcharge is turned off automatically. Defaults to: Yes.

- Added OverchargeExhaustedMessage. Message to be Shown when the Unit has automatically turned off Overcharge. Defaults to: 'GUI:OverchargeExhausted'.

- PoisonedBehavior:

- ~~Added IsNotAbsoluteKill. This enables the poisoned unit to trigger Death Prevention Behaviors like 'UndeadBody' and 'HighlanderBody'.~~

  Bug Fixed by hanfield.

- Memory Pool increased from {512,64} to {512, 128}.
- Expanded customizability with the hardcoded mechanics, including determined DamageAmount, or having DamageTypes and DeathTypes other than POISON.

```
{ "PoisonDamage",                    INI::parseReal,                        NULL,
{ "PoisonDamageMultiplier",          INI::parseReal,                        NULL,
{ "PoisonDamageType",                DamageTypeFlags::parseSingleBitFromINI, NULL,
{ "PoisonDamageTypeFX",              DamageTypeFlags::parseSingleBitFromINI, NULL,
{ "PoisonDeathType",                 INI::parseQuotedAsciiString,           NULL,
```

- Added PoisonDamage. You are able to customize the Poison to deal a set amount of damage instead of depending on the damage output of the poison.
- Added PoisonDamageMultiplier. PoisonBehavior by default deals damage based on the Weapon that triggers the Poison. PoisonDamageMultiplier can customize the existing values based on a multiplier. Declared with a value instead of percentages (e.g. 3.3.) Not compatible with PoisonDamage, gets overridden if PoisonDamage is declared.
- Added PoisonDamageType, PoisonDamageTypeFX and PoisonDeathType. You can now customize the poison to do the declared damage types and customize the DamageTypeFX and DeathType outside of POISON damage.

```
{ "PoisonDamageStatusType",              ObjectStatusMaskType::parseSingleBitFromINI,    NULL,
{ "PoisonDoStatusDamageType",            INI::parseBool,                                 NULL,
{ "PoisonStatusDuration",                INI::parseReal,                                 NULL,
{ "PoisonStatusDurationDamageCorrelation", INI::parseBool,                               NULL,
{ "PoisonStatusTintStatus",              TintStatusFlags::parseSingleBitFromINI,         NULL,
{ "PoisonStatusCustomTintStatus",        INI::parseQuotedAsciiString,                    NULL,
```

- Added PoisonDamageStatusType, PoisonDoStatusDamageType, PoisonStatusDuration, PoisonStatusDurationDamageCorrelation, PoisonStatusTintStatus, and PoisonStatusCustomTintStatus.

- All of which functions exactly as declared in the Weapon Template. Triggers every time the Poison triggers.
- PoisonStatusTintStatus defaults to: TINT_STATUS_POISONED.

```
{ "PoisonCustomDamageType",                INI::parseAsciiString,
{ "PoisonCustomDamageStatusType",          INI::parseAsciiString,
{ "PoisonCustomDeathType",                 INI::parseAsciiString,
```

- Added PoisonCustomDamageType, PoisonCustomDamageStatusType, and PoisonCustomDeathType, supporting CustomTypes features.

```
{ "DamageTypesReaction",                INI::parseDamageTypeFlagsCustom,        NULL,
{ "CustomDamageTypesReaction",          INI::parseCustomTypes,                  NULL,
```

- Added DamageTypesReaction, and CustomDamageTypesReaction. By default, PoisonedBehavior only triggers with DamageTypes that with POISON damage. Other than ‘WeaponIsPoison’ from the above feature configuration, you can customize the DamageTypes able to trigger the Poison. Defaults to: NONE +POISON. You are required to redeclare ‘ALL’ or ‘NONE’ when assigning the DamageTypes otherwise it would not work properly. They function similarly to DamageTypes of ‘FireWeaponWhenDamagedBehavior’ feature above.

```
{ "RequiredStatus",                 ObjectStatusMaskType::parseFromINI,     NULL,
{ "ForbiddenStatus",                ObjectStatusMaskType::parseFromINI,     NULL,
{ "RequiredCustomStatus",           INI::parseAsciiStringVector,            NULL,
{ "ForbiddenCustomStatus",          INI::parseAsciiStringVector,            NULL,
```

- Added RequiredStatus, ForbiddenStatus, RequiredCustomStatus, and ForbiddenCustomStatus. If declared, the Object requires to satisfy the Statuses to trigger the Behavior. Declared in the same way as SlowDeathBehavior or FireWeaponCollide.
- Note: The RequiredStatus, and RequiredCustomStatus requires having All of the declared statuses instead of any of the statuses, while ForbiddenStatus, and ForbiddenCustomStatus requires having Any of the declared statuses.

```
{ "DontCurePoisonOnHeal",           INI::parseBool,                         NULL,
{ "DamageTypesCurePoison",          INI::parseDamageTypeFlagsCustom,        NULL,
{ "CustomDamageTypesCurePoison",    INI::parseCustomTypes,                  NULL,
```

- Added DontCurePoisonOnHeal. On default, Poison is removed whenever an Object is Healed. Setting this to ‘Yes’ disables it. Defaults to: No.

- Added DamageTypesCure, and CustomDamageTypesCure. Which removes the Poison whenever the Object is hit with a DamageType. Defaults to: NONE.

- ProductionUpdate:

```
{ "ProductionModifier", parseAppendProductionModifier, NULL, offsetof( ProductionUpdateM
{ "BindsSelectionForGroupsProduced",     INI::parseBool, NULL, offsetof( ProductionUpdate
```

- Added ProductionModifier. An extended version of QuantityModifier. Declared in the same Format as QuantityModifier, but able to Register more Objects Behind. Objects Further registered are Produced in the Same Line as the First Object registered. This Production Bypasses Prerequisites and MaxSimultaneousOfType limit.

  NOTE: If the Declared Object clashes with QuantityModifier, it will Override the value registered by QuantityModifier.
  Defaults to: Not Configured.

  For example,

  ProductionModifier = ChinaInfantryTankHunter 5 ChinaInfantryRedguard 2 ChinaTankOverlord 1 GLAInfantryHijacker 3

  If declaring the above parameters within ProductionUpdate will cause the Producer to produce 5 ChinaInfantryTankHunters, 2 ChinaInfantryRedguards, 1 ChinaTankOverlord, and 3 GLAInfantryHijackers after finish Producing a ChinaInfantryTankHunter.

- Added BindsSelectionForGroupsProduced. If this is Configured to ‘Yes’, when Selecting any Production from the Object that produces out of ProductionModifier or QuantityModifier, all of the Units from the same Production Line are Simultaneously Selected. This also goes for Deselection. Defaults to: No.

- StealthUpdate:

```
{ "RequiredCustomStatus",   INI::parseAsciiStringVector, NULL,  offsetof( StealthUpdateModuleData,
{ "ForbiddenCustomStatus",  INI::parseAsciiStringVector, NULL,  offsetof( StealthUpdateModuleData,
```

- Added RequiredCustomStatus, and ForbiddenCustomStatus. Enable Custom Statuses to work with StealthUpdate. Functions the same way as existing RequiredStatus and ForbiddenStatus functions.

> Disguises:

- Disguise Feature has undergone a heavy rework, enabling it to be compatible with Weapons and Firing Offsets.

- Disguises are also extended with various features that enable it to be more viable. One of the examples are disguising as different Kindofs other than VEHICLES (See SpecialAbilityUpdate.)

```
{ "InnateDisguise",                          INI::parseBool,
{ "AutoDisguiseWhenAvailable",               INI::parseBool, NULL, offsetof( StealthUpdate
{ "CanStealthWhileDisguised",                INI::parseBool, NULL, offsetof( StealthUpdate
{ "DisguiseRetainAfterDetected",             INI::parseBool, NULL, offsetof( StealthUpdate
{ "PreservePendingCommandWhenDetected",      INI::parseBool, NULL, offsetof( StealthUpdate
```

- Added InnateDisguise. Enable the Unit to Instantly Disguise the moment they are Spawned. Does nothing if there are no existing Objects that the Unit can Disguise as. Defaults to: No.
- Added AutoDisguiseWhenAvailable. When Enabled, the Unit does not need to use the GUI Command to select a new disguise, the Unit will switch back to the Template that it was last Disguised as. It uses ‘StealthDelay’ for determining the time it switches back to its last Disguise. Defaults to: No.
- Added CanStealthWhileDisguised. Currently Stealth and Disguise modules do not stack, and Units that Disguise cannot Stealth. This changes the configuration to enable Stealth features for Disguises. Defaults to: No.
- Added DisguiseRetainAfterDetected. When Detected by StealthDetectorUpdate, the Disguised Unit will have their Disguises removed. Setting this to ‘Yes’ however, does not remove the Disguised Template. However, Enemies are still able to Recognize your Disguise and will attack you on Spot if the Unit does not Re-Disguise or if StealthDelay for ‘AutoDisguiseWhenAvailable’ has not yet reached. Defaults to: No.
- Added PreservePendingCommandWhenDetected. When a Disguised Unit has been Detected by StealthDetectorUpdate while it is Selected, the current pending GUI Command will be interrupted and removed. Setting this to ‘Yes’ disables that feature. Defaults to: No.

```
{ "DisguiseFlickerTransitionTime",  INI::parseDurationUnsignedInt,  NULL, offsetof( StealthUpdate
{ "DisguiseFriendlyFlickerDelay",   INI::parseDurationUnsignedInt,  NULL, offsetof( StealthUpdate
```

```
{ "DontFlashWhenFlickering",           INI::parseBool, NULL, offsetof( StealthUpdateModuleData
```

- Added DisguiseFlickerTransitionTime. If ‘DisguiseFriendlyFlickerDelay’ is configured, this sets the Transition Time for each Flicker to occur. The Disguised Object instantly Flickers if ‘0’ is configured. Defaults to: 0.
- Added DisguiseFriendlyFlickerDelay. If configured, enables Disguised Objects to briefly Reveal Themselves to Allies their Original Model after a configured Delay. The Object then switches back to its Disguise after its Reveal when another

Delay has reached. This alters between each Flicker. Does nothing towards Players that are not the Object's Allies. Defaults to: 0.

- Added DontFlashWhenFlickering. Objects that Flicker will have a White Flash briefly happen on them. This happens because the Object is currently selected, and the flash occurs during the Reselecting process of the Flickering. Setting this to 'Yes' disables the Flash from occurring while Flickering. Defaults to: No.

```
{ "UseOriginalFiringOffsetWhileDisguised",      INI::parseBool, NULL, offsetof( StealthUpda
{ "IsSimpleDisguiseForWeapons",     INI::parseBool, NULL, offsetof( StealthUpdateModuleData
```

- Added UseOriginalFiringOffsetWhileDisguised. If the Object can Attack while Disguised, it will attempt to use its Original Offset while Attacking. By default, the Object will use the Disguised Object's Offsets and Bones for determining its Firing Position. Defaults to: No.
- Added IsSimpleDisguiseForWeapons. If this is Set to 'Yes', Projectiles fired while Disguised will not consider any Bones and Offsets while firing. Defaults to: No.

- StealthDetectorUpdate:

```
{ "TriggeredBy",                    INI::parseAsciiStringVector,            NULL,
{ "ConflictsWith",                  INI::parseAsciiStringVector,            NULL,
```

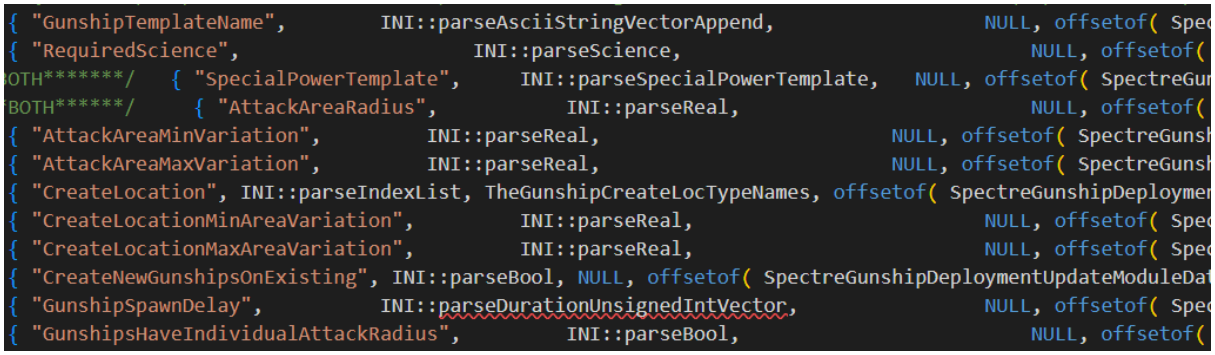
- Added TriggeredBy. and ConflictsWith. Enable Upgrade to work with StealthDetectorUpdate. If Upgrades are declared, StealthDetectorUpdate is required to meet the Upgrade Conditions to be able to function.

```
{ "ExtraRequiredStatus",              ObjectStatusMaskType::parseFromINI, NULL, offsetof( S
{ "ExtraForbiddenStatus",             ObjectStatusMaskType::parseFromINI, NULL, offsetof( S
{ "ExtraRequiredCustomStatus",  INI::parseAsciiStringVector, NULL,  offsetof( StealthDetector
{ "ExtraForbiddenCustomStatus", INI::parseAsciiStringVector, NULL,  offsetof( StealthDetector
```

- Added ExtraRequiredStatus. Units within detection range are required to have All of the declared statuses to be able to be Detected. Defaults to: None.
- Added ExtraForbiddenStatus. Units within detection range that has Any of the declared Statuses cannot be Detected by this Module. Defaults to: None.
- Added ExtraRequiredCustomStatus, and ExtraForbiddenCustomStatus. Enable Custom Statuses to work with StealthDetectorUpdate. Functions the same way as ExtraRequiredStatus and ExtraForbiddenStatus functions.

- SpectreGunshipDeploymentUpdate:

- This module has been modified to allow the deployment of multiple Spectre Gunships per Special Power Trigger. Refer to: here for practical demonstration with INI coding.



- GunshipTemplateName has been modified to allow the spawning of multiple Spectre Gunships per session.
- Added AttackAreaMinVaration. Configures the minimum varying distance from the Target Area for Spectre Gunships spawned by the Special Power after its first one. Defaults to: 0.
- Added AttackAreaMaxVaration. Configures the maximum varying distance from the Target Area for Spectre Gunships spawned by the Special Power after its first one. If AttackAreaMaxVaration is smaller than AttackAreaMinVaration, then the system will use AttackAreaMinVaration's value for its AttackAreaMaxVaration. Defaults to: 0.
- Added CreateLocationMinAreaVaration. Configures the minimum varying spawning distance of the spawning location for Spectre Gunships spawned by the Special Power after its first one. Defaults to: 0.
- Added CreateLocationMaxAreaVaration. Configures the maximum varying spawning distance of the spawning location for Spectre Gunships spawned by the Special Power after its first one. If CreateLocationMaxAreaVaration is smaller than CreateLocationMinAreaVaration, then the system will use AttackAreaMinVaration's value for its CreateLocationMaxAreaVaration. Defaults to: 0.
- Added GunshipSpawnDelay. Sets the spawning delay for gunships configured within 'GunshipTemplateName. Respective Gunships will use the respective values configured after each spacing to be parsed as vector. Defaults to: 0 or no delay.
- Added GunshipsHaveIndividualAttackRadius. A new mechanic added, if this is not set, then all gunships spawned will only have varying Targeting Position configured by Attack Area Variation but within the same Attack Area. If this is set

to ‘Yes’, each individual gunship will spawn with their own Attack Area for their own Attack Area Variation. Defaults to: No.

- SpectreGunshipUpdate:

- Multiple Spectre Gunships can be spawned without requiring the modification of Source Code. Refer to: here for practical demonstration with INI coding.

```
{ "UseMyProducerToInitiateSpecialPower", INI::parseBool,          NULL, offsetof( SpectreGunshi
{ "UseMyLocomotorToOrbit",               INI::parseBool,          NULL, offsetof( SpectreGunshi
{ "PlaySoundOnCreationWithSpawnDelay",   INI::parseBool,          NULL, offsetof( SpectreGunshi
{ "CursorName",                    INI::parseAsciiString,                   NULL, offsetof(
```

- Added UseMyProducerToInitialSpecialPower. Currently the feature can be implemented by declaring ScriptedSpecialPowerOnly = Yes onto the Object’s ‘SpecialPower’ Module. But this can also be declared as an alternative. Defaults to: No.

- Added UseMyLocomotorToOrbit. A different approach to SpectreGunship Orbiting Behavior. By default all Jets have the ability to Circle if they are left Idle, including SpectreGunship. The default module uses a calculation every frame to determine the next position the gunship should orbit to. This variable enables the Gunship to use its Locomotor to Orbit.

  ~~If this is enabled, the gunship is first required to fly to the targeted area instead of approaching its destination for orbiting. ‘GunshipOrbitRadius’ is obsolete and the Object uses the SET_NORMAL Locomotor’s CirclingRadius for its OrbitingRadius.~~
  Update: Now uses GunshipOrbitRadius to determine its Initial destination. After reaching the destination it will orbit around the Center with its Orbiting Locomotor (SET_NORMAL’s) CirclingRadius.

  Defaults to: No.

- Added PlaySoundOnCreationWithSpawnDelay. Due to the usage of a function within the SpectreGunshipUpdate playing its sound when setting the Attack Area Variation with GunshipsHaveIndividualAttackRadius set to ‘No’ (not GunshipSpawnDelay). If this is enabled, then the ‘VoiceMove’ of the Gunship will be played when the above conditions are met. Defaults to: No.

- Added CursorName. Configures the Cursor Name when selecting the Spectre Gunship. If it is not configured, then it uses the default ‘ParticleUplinkCannon’ Mouse Template. Defaults to: Not configured.

Player:

- PlayerTemplate:

```
{ "SelectingCursorName",              INI::parseAsciiString,
{ "ArrowCursorName",                  INI::parseAsciiString,
{ "ScrollCursorName",                 INI::parseAsciiString,
{ "MoveCursorName",                   INI::parseAsciiString,
{ "MoveInFormationCursorName",        INI::parseAsciiString,
{ "AttackMoveCursorName",             INI::parseAsciiString,
{ "WaypointCursorName",               INI::parseAsciiString,
{ "GenericInvalidCursorName",         INI::parseAsciiString,
{ "EnterCursorName",                  INI::parseAsciiString,
{ "EnterAggressiveCursorName",        INI::parseAsciiString,
{ "TargetCursorName",                 INI::parseAsciiString,
{ "AttackCursorName",                 INI::parseAsciiString,
{ "ForceAttackCursorName",            INI::parseAsciiString,
{ "ForceAttackGroundCursorName",      INI::parseAsciiString,
{ "OutrangeCursorName",               INI::parseAsciiString,
{ "GetRepairAtCursorName",            INI::parseAsciiString,
{ "DockCursorName",                   INI::parseAsciiString,
{ "GetHealedCursorName",              INI::parseAsciiString,
{ "DoRepairCursorName",               INI::parseAsciiString,
{ "ResumeConstructionCursorName",     INI::parseAsciiString,
{ "SetRallyPointCursorName",          INI::parseAsciiString,
{ "BuildCursorName",                  INI::parseAsciiString,
{ "InvalidBuildCursorName",           INI::parseAsciiString,
```

- Added "SelectingCursorName". Changes Selecting Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "ArrowCursorName". Changes Arrow Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "ScrollCursorName". Changes Scroll Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "MoveCursorName". Changes Move Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "MoveInFormationCursorName". Changes Move In Formation Cursor for this Player Template if configured. Defaults to: Not Configured
- Added "AttackMoveCursorName". Changes Attack Move Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "WaypointCursorName". Changes Waypoint Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "GenericInvalidCursorName". Changes GenericInvalid Cursor for this Player Template if configured. Defaults to: Not Configured.

- Added "EnterCursorName". Changes Enter Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "EnterAggressiveCursorName". Changes EnterAggressive Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "TargetCursorName". Changes Target Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "AttackCursorName". Changes Attack Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "ForceAttackCursorName". Changes Force Attack Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "ForceAttackGroundCursorName". Changes Force Attack Ground Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "OutrangeCursorName". Changes Outrange Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "GetRepairAtCursorName". Changes Get Repair At Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "DockCursorName". Changes Dock Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "GetHealedCursorName". Changes Get Healed Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "DoRepairCursorName". Changes Do Repair Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "ResumeConstructionCursorName". Changes Resume Construction Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "SetRallyPointCursorName". Changes Set Rally Point Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added "BuildCursorName". Changes Build Cursor for this Player Template if configured. Defaults to: Not Configured.
- Added"InvalidBuildCursorName". Changes Invalid Build Cursor for this Player Template if configured. Defaults to: Not Configured.

Upgrades:

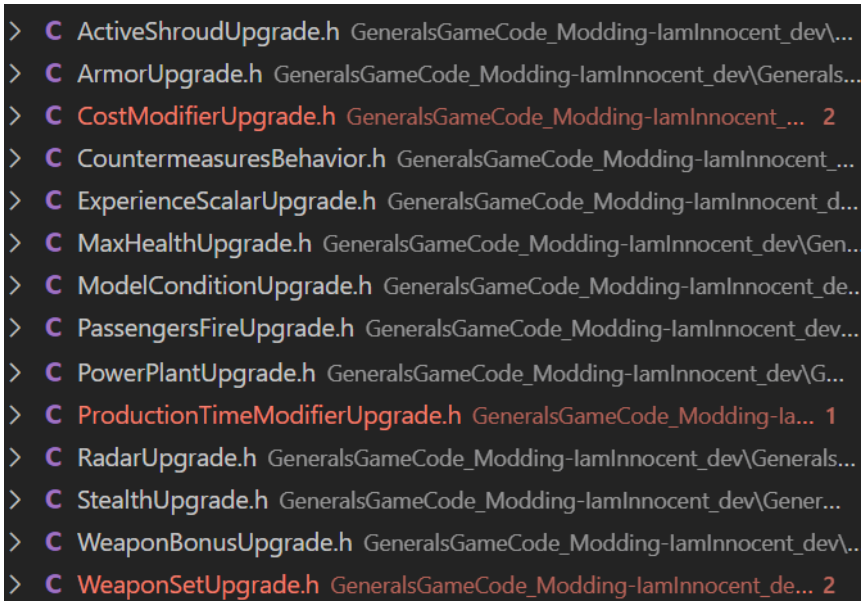
- Upgrade Modules:

```
{ "TriggeredBy",         INI::parseAsciiStringVector, NULL, offsetof( UpgradeMuxData, m_activationUpgradeNames ) },
{ "ConflictsWith",   INI::parseAsciiStringVector, NULL, offsetof( UpgradeMuxData, m_conflictingUpgradeNames ) },
{ "RemovesUpgrades",INI::parseAsciiStringVector, NULL, offsetof( UpgradeMuxData, m_removalUpgradeNames ) },
{ "GrantUpgrades",       INI::parseAsciiStringVector, NULL, offsetof( UpgradeMuxData, m_grantUpgradeNames ) },
{ "FXListUpgrade",   INI::parseFXList, NULL, offsetof( UpgradeMuxData, m_fxListUpgrade ) },
{ "RequiresAllTriggers", INI::parseBool, NULL, offsetof( UpgradeMuxData, m_requiresAllTriggers ) },
```

- Added "GrantUpgrades" that uses inputs exactly like RemovesUpgrades, but instead grants upgrades to the Player or Object upon activation.
- GrantUpgrades consider whether the Upgrade is a Player Upgrade or an Object Upgrade before Granting them.
- Player Upgrade grants to the player normally as when the player completes an upgrade.
- Object Upgrade grants the upgrade to the object itself.

- (Experimental) RemovesUpgrades can now also remove player upgrades.
- This feature may induce many chaotic elements and bugs due to how Player Upgrades interact with Existing Objects.
- After a Player Upgrade has been removed, the Player requires to purchase the Upgrade Again to Grant Bonuses.

- There are several Upgrades that do nothing when they are removed even though they grant bonuses when Upgraded. Several of them have been implemented with Upgrade Removal features. These include:



- These upgrades revert their values back to previous state when the Upgrade is removed:

> ActiveShroudUpgrade
> CostModifierUpgrade
> ExperienceScalarUpgrade
> MaxHealthUpgrade
> PowerPlantUpgrade
> ProductionTimeModifierUpgrade

- These upgrades remove their flags or Statuses after the Upgrade is removed:

> ArmorUpgrade
> ModelConditionUpgrade
> StealthUpgrade (reverted Stealth grant/ remove for Modding Devs' Implemented Feature)
> WeaponBonusUpgrade
> WeaponSetUpgrade

- These upgrades disables their functionality after the Upgrade is removed:

> CountermeasuresBehavior
> LineOfSightModifierUpgrade
> PassengersFireUpgrade
> RadarUpgrade

- GrantUpgradeCreate And UpgradeDie:

```
{ "UpgradesToGrant",     INI::parseAsciiStringVector,
{ "UpgradesToRemove",    INI::parseAsciiStringVector,
```

- Added both features of "UpgradesToGrant" and "UpgradesToRemove" which function similarly to "GrantUpgrades" and "RemovesUpgrades".
- "UpgradesToRemove" and "GrantUpgrades" for UpgradeDie considers it's 'Master' or the Player for Upgrade Removal or Granting after dying.

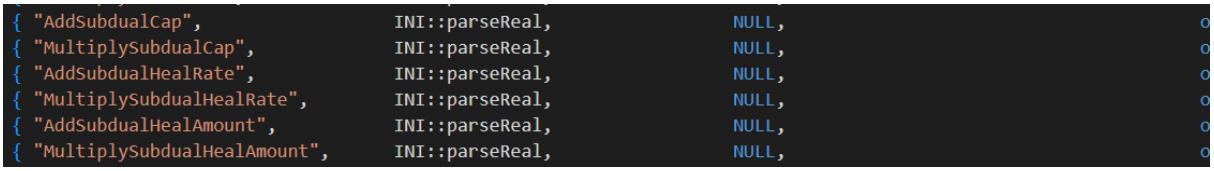
- NEW LineOfSightModifierUpgrade:

- A new UpgradeModule that enables the Modification for Requiring Line of Sight for a Unit. Removing Upgrade removes the Disability.

```
{ "DisableNeedToCheckLineOfSight", INI::parseBool, NULL, offsetof(LineOfSightModifier
{ "SetIgnoreObstacleViewBlock", INI::parseBool, NULL, offsetof(LineOfSightModifierUpg
```

- Added DisableNeedToCheckLineOfSight. If the Module is Active, setting this to ‘Yes’ disables the need for the Object to Check Line of Sight for Moving or Attacking. Defaults to: No.
- Added SetIgnoreObstacleViewBlock. If the Module is Active, setting this to ‘Yes’ disables the need for the Object to check for Stationary or Blocking Targets for Moving or Attacking. Defaults to: No.

- Max Health Upgrade:

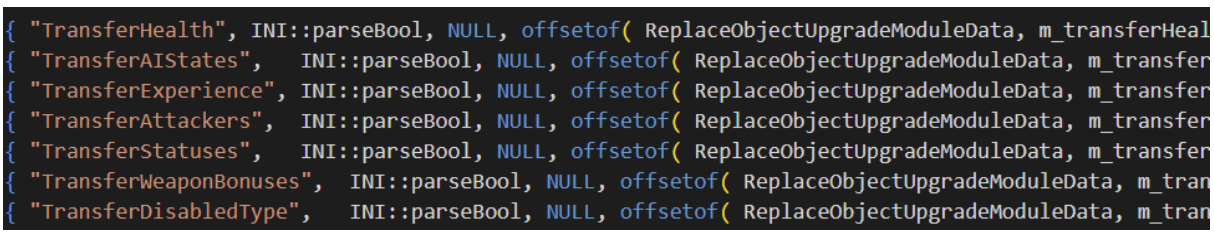


- Added AddSubdualCap, MultiplySubdualCap, AddSubdualHealRate, MultiplySubdualHealRate, AddSubdualHealAmount, MultiplySubdualHealAmount.
- You can now also customize the Subdual Cap, Subdual Heal Rate, and Subdual Heal Amount when applying Max Health Upgrade module.
- Subdual properties are not affected by ChangeType as they’re only applicable to MaxHealth changes.
- CustomSubdualDamageCap, CustomSubdualDamageHealRate, and CustomSubdualDamageHeal changes according to the ratio according to the new SubdualDamage value to their old.

Note: If the Subdual Damage Cap is multiplied to ‘0’. It will make the upgraded Objects Immune to Subdual Damage.

- ReplaceObjectUpgrade:

- Added various configuring Variables for Objects Spawned.



- Added "TransferHealth". Does the Object transfer its Health onto the Spawned Object? The Health Inherit Type configuration is configured with ‘HealthTransferType’ Variable. Defaults to: No.

- Added "TransferAIStates". Does the Object transfer its AI States (Moving, Attacking, Constructing, Repairing) unto the Object Created? Defaults to: No.
- Added "TransferExperience". Does the Object transfer its Experience Points unto the Object Created? Defaults to: No.
- Added "TransferAttackers". Does the Module tell recent Attackers of the Object to attack the Object Created? Defaults to: No.
- Added "TransferStatuses". Does the Object transfer its Statuses unto the Object Created? Defaults to: No.
- Added "TransferWeaponBonuses". Does the Object transfer its Weapon Bonuses unto the Object Created? Defaults to: No.
- Added "TransferDisabledType". Does the Object transfer its DisabledTypes unto the Object Created? Defaults to: No.

```
{ "TransferSelection",  INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData, m_transfer
{ "TransferSelectionDontClearGroup",    INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModule
{ "TransferObjectName", INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData, m_transfer
{ "HealthTransferType",     INI::parseIndexList,        TheMaxHealthChangeTypeNames, offsetof( Rep
```

- Added "TransferSelection". Is the Created Object configured to be Selected if the Object is Selected? Defaults to: No.
- Added "TransferSelectionDontClearGroup". Does ‘TransferSelection’, when Triggered, clear the current Selection Group? Defaults to: No.
- Added "TransferObjectName". Does the Object transfer ObjectName unto the Object Created? Enabling Scripts to identify the Created Object as the current Object. Defaults to: No.
- Added "HealthTransferType". Determines the Health Inherit Type for “TransferHealth” of the Module. Different Configuration Types:
  - PRESERVE_RATIO : Changes the Health according to the Health Ratio.
  - SAME_CURRENTHEALTH : Changes the Health to Match the Object’s Current Health.
  - ADD_CURRENT_DAMAGE : Changes the Missing Health to the Object’s Current Missing Health (Can kill if Missing Health exceeds Max Health.)
  - ADD_CURRENT_DAMAGE_NON_LETHAL: Changes the Missing Health to the Object’s Current Missing Health (If Missing Health exceeds Max Health, Object’s Health is set to 1.)

- Defaults to: SAME_CURRENTHEALTH.

```
{ "TransferBombs",  INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData, m_transferBo
{ "TransferHijackers",  INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData, m_transf
{ "TransferEquippers",  INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData, m_transf
{ "TransferParasites",  INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData, m_transf
{ "TransferPassengers", INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData, m_transf
{ "TransferToAssaultTransport", INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData,
{ "TransferShieldedTargets",    INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData,
{ "TransferShieldingTargets",   INI::parseBool, NULL, offsetof( ReplaceObjectUpgradeModuleData,
```

- Added "TransferBombs". If Set to ‘Yes’, transfer any Sticky Bombs from the Object that Triggers the Module unto the Spawned Object. Defaults to: No.
- Added "TransferHijackers". If Set to ‘Yes’, transfer any Hijackers from the Object that Triggers the Module unto the Spawned Object. Defaults to: Yes.
- Added "TransferEquippers". If Set to ‘Yes’, transfer any Equippers, excluding Parasites, from the Object that Triggers the Module unto the Spawned Object. Defaults to: Yes.
- Added "TransferParasites". If Set to ‘Yes’, transfer any Parasites from the Object that Triggers the Module unto the Spawned Object. Defaults to: Yes.
- Added "TransferPassengers". If Set to ‘Yes’, transfer any Passengers from the Object that Triggers the Module unto the Spawned Object. If not, the Passengers will be Ejected. Defaults to: Yes.
- Added "TransferToAssaultTransport". If Set to ‘Yes’, transfer the Object’s Assault Transport status onto the Spawned Object and Remove it from Assault Transport. Defaults to: No.
- Added "TransferShieldedTargets". If Set to ‘Yes’, transfer the Spawner’s SHIELDED_TARGET Status to the Spawned Objects along with the Shielder’s information and ProtectionTypes. Status and Information is then removed from the Spawner. Defaults to: No.
- Added "TransferShieldingTargets". If Set to ‘Yes’, transfer the Spawner’s Current Shielding Target’s Information to the Spawned Objects. Shielding Information is then removed from the Spawner. Defaults to: No.

```
{ "OrientInForceDirection", INI::parseBool, NULL, offsetof(ReplaceObjectUpgradeModuleData, m_ori
{ "ExtraBounciness",               INI::parseReal,                    NULL, offsetof( ReplaceO
{ "ExtraFriction",             parseFrictionPerSec,                       NULL, offsetof( Repl
{ "Offset",                    INI::parseCoord3D,             NULL, offsetof( ReplaceObjectUpg
{ "Disposition",           INI::parseBitString32,         DispositionNames, offsetof( ReplaceO
{ "DispositionIntensity",   INI::parseReal,                    NULL,   offsetof( ReplaceObjectU
{ "SpinRate",                  INI::parseAngularVelocityReal,  NULL, offsetof(ReplaceObjectUpgr
{ "YawRate",                   INI::parseAngularVelocityReal,  NULL, offsetof(ReplaceObjectUpgr
{ "RollRate",                  INI::parseAngularVelocityReal,  NULL, offsetof(ReplaceObjectUpgr
{ "PitchRate",             INI::parseAngularVelocityReal,  NULL, offsetof(ReplaceObjectUpgradeM
{ "MinForceMagnitude",  INI::parseReal, NULL, offsetof(ReplaceObjectUpgradeModuleData, m_minMag)
{ "MaxForceMagnitude",  INI::parseReal, NULL, offsetof(ReplaceObjectUpgradeModuleData, m_maxMag)
{ "MinForcePitch",  INI::parseAngleReal,    NULL, offsetof(ReplaceObjectUpgradeModuleData, m_min
{ "MaxForcePitch",  INI::parseAngleReal,    NULL, offsetof(ReplaceObjectUpgradeModuleData, m_max
{ "DiesOnBadLand",  INI::parseBool, NULL, offsetof(ReplaceObjectUpgradeModuleData, m_diesOnBadLa
```

- Added ObjectCreationList's Disposition and Spawning Variables onto ReplaceObjectUpgrade Module.

- Status Bits Upgrade (Modding Devs' Implemented Feature):

```
{ "StatusToSet",        ObjectStatusMaskType::parseFromINI, NULL, offsetof( StatusBitsUpgradeModu
{ "StatusToClear",  ObjectStatusMaskType::parseFromINI, NULL, offsetof( StatusBitsUpgradeModuleDa
{ "CustomStatusToSet",  INI::parseAsciiStringVector, NULL, offsetof( StatusBitsUpgradeModuleData,
{ "CustomStatusToClear",    INI::parseAsciiStringVector, NULL, offsetof( StatusBitsUpgradeModuleD
{ "BonusToSet",       INI::parseWeaponBonusVector, NULL, offsetof( StatusBitsUpgradeModuleData, m
{ "BonusToClear",       INI::parseWeaponBonusVector, NULL, offsetof( StatusBitsUpgradeModuleData,
{ "CustomBonusToSet",   INI::parseAsciiStringVector, NULL, offsetof( StatusBitsUpgradeModuleData,
{ "CustomBonusToClear", INI::parseAsciiStringVector, NULL, offsetof( StatusBitsUpgradeModuleData,
```

- Added CustomStatusToSet, and CustomStatusToClear, supporting the usage for CustomStatusTypes.
- Added BonusToSet, and BonusToClear. Now also supports granting Weapon Bonus Types. Weapon Bonus granted by StatusBitsUpgrade will not get removed by FiringTrackerBonuses, but will get overridden by WeaponBonusUpdate.
- Added CustomBonusToSet, and CustomBonusToClear. Supporting Custom Weapon Bonus Types.

The features listed are an Alpha version which majority of them have been tested and are working. Their stability in a full game however, have not been tested and require testing.