

# Custom Objective Tutorial

This tutorial demonstrates the creation and use of custom objectives. These objectives will be similar to the `Zen_CreateObjective` function. Custom objectives are greatly simplified by the Co-op Framework's Task subsystem: a set of functions that make it easy for a scenario creator to create, update and terminate game engine tasks.

To complete this tutorial you will modify the mission's init script. The map elements needed for the tutorial are supplied.

## Preparing to Create the Mission

Copy the `CustomObjectives.Altis` subdirectory to:

```
<MyDocuments>\Arma 3 - Other Profiles\<MyProfile>\missions\
```

Copy the `Zen_FrameworkFunctions` directory from the `Shell.Stratis` directory to:

```
<MyDocuments>\Arma 3 - Other Profiles\<MyProfile>\missions\ CustomObjectives.Altis\
```

## Review the Map

Open Arma and then the Editor. Select Altis Island and Continue.

Choose the load function and then select `CustomObjectives`.

Northeast of Kavala are two partisans and the unit you will play. This unit, X11, is 'parked' near the Stadium. The other two partisans are also 'parked' near the stadium but are not part of the squad. The mission logic will move the partisans to a random position in the West. As the squad leader you will rendezvous with your fellow partisans and a single squad will be created.

No changes are required to the map.

## Updating the Initialization Script – Part One

Open the `init.sqf` file in

```
<MyDocuments>\Arma 3 - Other Profiles\<MyProfile>\missions\ CustomObjectives.Altis\
```

Bump up the Title Text wait time a bit.

```
titleText ["Good Luck", "BLACK FADED", 0.4];
```

Immediately above the lines

```
// All clients stop executing here, do not delete this line  
if (!isServer) exitWith {};
```

Add a mission briefing statement:

```
Player creatediaryRecord["Diary", ["Custom Objectives Tutorial",  
"Complete the tasks in the order presented.<br/>Designed for a single  
player led squad.<br/>"]];
```

At the point labeled 'Enter the mission code here' enter these statements:

Make the area markers invisible:

```
"RendezvousOne" setMarkerAlpha 0;  
"ClearArea" setMarkerAlpha 0;
```

Set the weather and time of day with a short 'sleep' to give minute/hour update time to 'stick':

```
0 = [["overcast", random 0.3, random 0.7, 60*45], ["fog", random 0.3,  
random 0.3, 60*15], ["date", random 60, 9 + random 7]] spawn  
Zen_SetWeather;  
sleep 1.0;
```

Create references to the squad leader (X11) and the two partisan units, concatenate them (as units) into a single array and call the typical 'helper' functions:

```
_group_X11 = group X11;  
_group_Partisan = group PartisanReinforcement;  
  
_allplayersArray = [_group_X11, _group_Partisan] call  
Zen_ConvertToObjectArray;  
  
0 = [_allplayersArray] call Zen_AddGiveMagazine;  
0 = [_allplayersArray] call Zen_AddRepackMagazines;  
0 = [_allplayersArray, "infantry"] call Zen_SetAISkill;  
0 = [_allplayersArray] call Zen_TrackInfantry;
```

## CREATE AN INLINE FUNCTION

Now enter the inline function that will create a mission objective. An objective is composed essentially of two things: a task and the triggers that complete the task.

Immediately beneath these lines:

```
// All clients stop executing here, do not delete this line  
if (!isServer) exitWith {};
```

Enter the function declaration:

```
f_createRendezvousObjective = {
};
```

Now enter the code for the custom objective. It will create a task that will describe what is required for the player to complete the task successfully, the position on the map where this task can be completed and two triggers for completing the task.

First, declare the private variables:

```
private ["_pUnits", "_pPosition", "_taskName", "_aredeadHandle"];
```

Retrieve the two parameters that will be passed to this function and assign them to local variables:

```
_pPosition = _this select 0;
_pUnits = _this select 1;
```

Now call a Framework function to create the task:

```
_taskName = [_pUnits, "Reach the Rendezvous point, collect your squad
and await further instructions", "Rendezvous", _pPosition] call
Zen_InvokeTask;
```

A task can be completed in three ways. It can be cancelled, the players assigned the task can fail or the players assigned the task will succeed.

The scenario designer uses the Co-op Framework Trigger functions to complete the task.

The first trigger sets the task to 'Failed' if the player is killed before completing the task. Obviously, the player is not going to be killed driving 50 yards in a Humvee but is included for completeness. Note that this trigger function is 'spawned'; the trigger will run in the background on its own thread.

```
_aredeadHandle = [_pUnits, _taskName, "failed"] spawn
Zen_TriggerAreDead;
```

The second trigger completes the task by changing its status to 'Successful'. The Co-op Framework trigger function allows the designer to define 'success' by setting the values of the function parameters:

```
0 = [_pUnits, _taskName, "succeeded", _pPosition, "one", 25] spawn
Zen_TriggerAreNear;
```

Return the 'are dead' handle. The mainline of init function will reference this.

```
(_aredeadHandle)
```

The function definition is complete.

## INVOKE THE INLINE FUNCTION

Scroll down to immediately after the 'helper functions' enter the code to create the rendezvous point, move the partisans to that point and create the player task:

```
_X11RendezvousOne = ["RendezvousOne"] call Zen_FindGroundPosition;  
0 = [_group_Partisan,_X11RendezvousOne] call Zen_MoveAsSet;  
  
_alldeadHandle = [_X11RendezvousOne, _group_X11] call  
f_createRendezvousObjective;  
  
sleep 1;  
  
_X11Task = [X11] call Zen_GetUnitTasks;  
waituntil { sleep 5; [_X11Task] call Zen_AreTasksComplete };
```

After the task is completed, merge the three units into a single squad led by X11 and terminate the trigger for task failure:

```
(units _group_Partisan) join _group_X11;  
terminate _alldeadHandle;
```

Save the init method and preview the mission.

## Updating the Initialization Script – Part Two

The second mission objective will require the player to clear an area of opposition forces.

## CREATE ANOTHER INLINE FUNCTION

Immediately after the Create Rendezvous function declare the second objective function declaration:

```
f_createClearAreaObjective= {  
};
```

First, declare the private variables:

```
private ["_pPosition", "_pUnits", "_pSide",  
"_pNumberOfSquads", "_pAverageSize", "_pSkillLevel", "_minSize", "_maxSize",  
"_oppositionArray", "_randomPosition", "_randomGroup", "_taskName", "_alldeadHandle"];
```

Retrieve the four required parameters and assign them to local variables:

```
_pPosition = _this select 0;  
_pUnits = _this select 1;  
_pSide = _this select 2;  
_pNumberOfSquads = _this select 3;
```

This function has two optional parameters. If they are not passed into the function then the function designer needs to assign reasonable default values to them. If the scenario designer needs the sixth parameter then the fifth must be supplied.

The fifth parameter is the average size of the opposition squad; the sixth is their general skill levels.

```
if (count _this > 4) then {
    _pAverageSize = _this select 4;
} else {
    _pAverageSize = 3;
};

_minSize = _pAverageSize - 2;
_maxSize = _pAverageSize + 2;
if (_minSize <= 0) then {_minSize = 1};

if (count _this > 5) then {
    _pSkillLevel = _this select 5;
} else {
    _pSkillLevel = "infantry";
};
```

Now create the squads to be 'cleared'. Of course, the designer of the function has as much control as the game engine and the Framework will allow: number of squads, skill levels, squad behavior, etc.

```
_oppositionArray = [];

for "_i" from 0 to (_pNumberofSquads - 1) do {
    _randomPosition = [_pPosition] call Zen_FindGroundPosition;
    _randomGroup = [_randomPosition, _pSide, _pSkillLevel,
    [_minSize, _maxSize]] call Zen_SpawnInfantry;
    _oppositionArray set [(count _oppositionArray), _randomGroup];
};

0 = [_oppositionArray, _pPosition] spawn Zen_OrderInfantryPatrol;
0 = [(_oppositionArray), "group"] call Zen_TrackInfantry;
```

Finally create the task, the fail trigger, the success trigger and then return the handle to the 'failure' trigger:

```
_taskName = [_pUnits, "Clear the designated area and await further
instructions", "Clear and Hold", _pPosition] call Zen_InvokeTask;

_aredeadHandle = [_pUnits, _taskName, "failed"] spawn
Zen_TriggerAreDead;
0 = [_oppositionArray, _taskName, "succeeded", _pPosition] spawn
Zen_TriggerAreaClear;

(_aredeadHandle)
```

## INVOKE THE SECOND INLINE FUNCTION

Finally, integrate this new objective into the mainline of the init function:

```
_alldeadHandle = ["ClearArea",_allplayersArray,EAST,4,3,"infantry"]  
call f_createClearAreaObjective;  
sleep 1;  
_X11Task = [X11] call Zen_GetUnitTasks;  
waituntil { sleep 5; [_X11Task] call Zen_AreTasksComplete };  
terminate _alldeadHandle;
```

Save the init method and preview the mission.

## Technical Corner

Extend this mission to be a co-op mission in which two 'real life' players each command separate squads and are assigned different objectives.

First, add a second playable squad leader:




Select X11. Copy with CTRL-C. Paste and 'park' nearby with CTRL-V.



Change the name of 'pasted' unit to X12 and make playable:


EDIT UNIT

B\_G\_Soldier\_F

SIDE:  BLUFOR

FACTION:  FIA

CLASS: Men


UNIT:  Rifleman

by Bohemia Interactive

SPECIAL: In Formation

CONTROL:  Playable

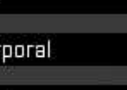
VEHICLE LOCK: Default

RANK:  Corporal

INFO AGE: Unknown

AZIMUTH: -41.57°


ELEVATION: 0°




NAME: X12


INITIALIZATION:

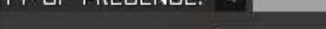
DESCRIPTION:

SKILL: <  >

HEALTH/ARMOR <  >

FUEL <  >

AMMUNITION <  >

PROBABILITY OF PRESENCE: <  >

CONDITION OF PRESENCE: true

PLACEMENT RADIUS: 0

CANCEL

OK

Select both units of PartisanReinforcement and make a copy of them. Set them playable.

Rename the new squad:

**EDIT UNIT** B\_G\_Soldier\_AR\_F

SIDE: BLUFOR

FACTION: FIA ▼

CLASS: Men ▼

UNIT: Autorifleman ▼

by Bohemia Interactive


SPECIAL: In Formation ▼

CONTROL: Playable ▼

VEHICLE LOCK: Default ▼

RANK: Corporal ▼

INFO AGE: Unknown ▼

AZIMUTH:  

ELEVATION:

NAME:

INITIALIZATION:

DESCRIPTION:

SKILL: <  >

HEALTH/ARMOR <  >

FUEL <  >

AMMUNITION <  >

PROBABILITY OF PRESENCE: <  >

CONDITION OF PRESENCE:

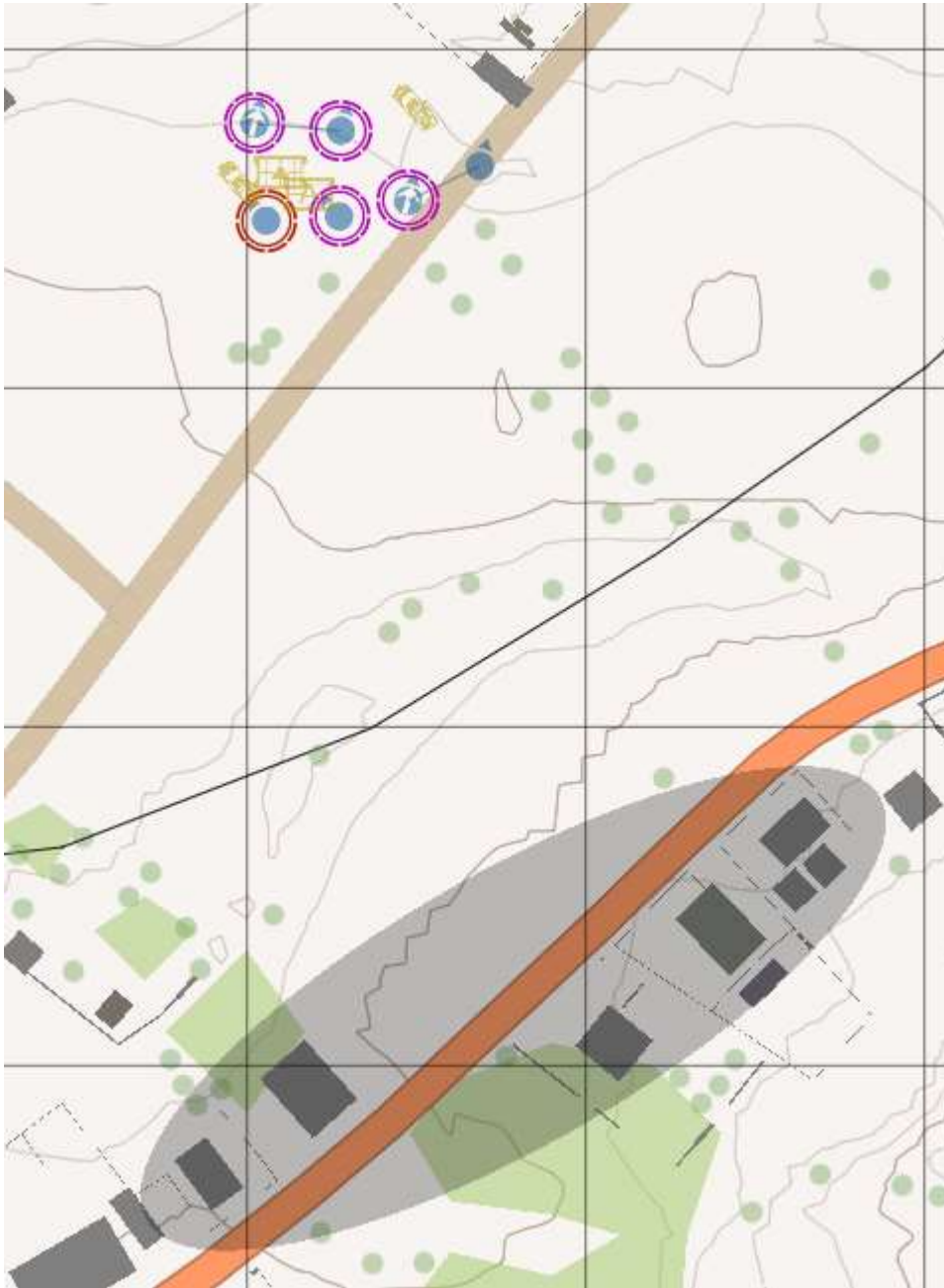
PLACEMENT RADIUS:

CANCEL OK

Copy and paste the off road vehicle so X12 squad has some transportation.



Create a second Rendezvous area to the South by copying and pasting Rendezvous One and then re-naming it 'RendezvousTwo'.



Key CTRL-S to save this map.

Remember, the original mission map for any tutorial can be found in [mission.sqm.org](http://mission.sqm.org), So experiment as much as you wish; you can always get back to 'square one' by copying and renaming the init and mission 'orig' files.

Preview the mission. You should see the three new units and their vehicle behind you. You've made no changes to the original single player init script so there should be no errors or warnings returned by game engine.

Open the init method for the Custom Objectives mission

Add the code to incorporate X12 and the player's reinforcements into the mission:

Clear new marker:

```
"RendezvousOne" setMarkerAlpha 0;  
"RendezvousTwo" setMarkerAlpha 0;  
"ClearArea" setMarkerAlpha 0;
```

Create references to new units:

```
_group_X11 = group X11;  
_group_Partisan = group PartisanReinforcement;  
_group_X12 = group X12;  
_group_PartisanX12 = group PartisanReinforcementX12;
```

Concatenate all players into single array:

```
_allplayersArray = [_group_X11, _group_Partisan, _group_X12,  
_group_PartisanX12] call Zen_ConvertToObjectArray;
```

Override the default co-op view distance:

```
0 = [4000, 1200] call Zen_SetViewDistance;
```

Assign a rendezvous objective to X12:

Immediately after this line:

```
_X11Task = [X11] call Zen_GetUnitTasks;
```

Add this code:

```
_X12RendezvousTwo = ["RendezvousTwo"] call Zen_FindGroundPosition;  
0 = [_group_PartisanX12, _X12RendezvousTwo] call Zen_MoveAsSet;  
_alldeadHandleX12 = [_X12RendezvousTwo, _group_X12] call  
f_createrendezvousObjective;  
sleep 1;  
_X12Task = [X12] call Zen_GetUnitTasks;
```

This tutorial mission as extended here was designed to be played with a second player – but will be coded so it is playable by single real life player. Add some behavior for X12:

```
if (!isplayer X12) then{  
    _group_X12 move _X12RendezvousTwo;  
};
```

Now the script needs logic to recognize that the rendezvous is complete and that the partisans can be joined to the squad leader and the “all dead” background task can be terminated.

Since the functional logic is the same for each squad put the code into a new inline function.

Immediately beneath *f\_createClearAreaObjective* enter this code:

```
f_isrendezvoustaskcomplete = {  
  
    private ["_pUnit", "_pTask", "_pPartisan", "_pHandle"];  
  
    _pUnit = _this select 0;  
    _pTask = _this select 1;  
    _pPartisan = _this select 2;  
    _pHandle = _this select 3;  
  
    waituntil { sleep 5; ([_pTask] call Zen_AreTasksComplete) };  
  
    (units _pPartisan) join _pUnit;  
    terminate _pHandle;  
};
```

Now scroll back to mainline of init.

Replace these lines:

```
waituntil { sleep 5; [_X11Task] call Zen_AreTasksComplete };  
(units _group_Partisan) join _group_X11;  
terminate _alldeadHandle;
```

With these:

```
rendezvousHandleX11 =  
[_group_X11, _X11Task, _group_Partisan, _alldeadHandle] spawn  
f_isrendezvoustaskcomplete;  
rendezvousHandleX12 =  
[_group_X12, _X12Task, _group_PartisanX12, _alldeadHandleX12] spawn  
f_isrendezvoustaskcomplete;  
  
waituntil { sleep 5; (scriptdone rendezvousHandleX11) && (scriptdone  
rendezvousHandleX12) };
```

The joining of partisans to squad leader and termination of failure trigger are now encapsulated in the *f\_isrendezvoustaskcomplete* function.

The call to create the ‘Clear and Hold’ objective follows this unchanged.

But after this line:

```
_X11Task = [X11] call Zen_GetUnitTasks;
```

Add code to give X12's squad default behavior and a little boost in survivability:

```
if (!isplayer X12) then{
    0 = [(units _group_X12), "SOF"] call Zen_SetAISkill;
    0 = [_group_X12, "ClearArea"] spawn Zen_OrderInfantryPatrol;
};
```

## Tasks in General

Even this extended edition of the tutorial mission does not explore all the potentialities of managing tasks.

Just beneath this line near the bottom of the init function:

```
_X11Task = [X11] call Zen_GetUnitTasks;
```

Put this:

```
player sidechat str _X11Task;
```

Then preview the mission past the completion of the Rendezvous objective.

The sidechat will display two strings in an array. The first string will be the name of the Rendezvous task; the second the name of the Clear and Hold task. As the tasks are added to a player they are appended to the end of the array.

If you coded a mission with multiple active tasks and you wanted to micro-manage “failure” and “success” then you could save off the names of tasks in global variables (or a global array) and manage them individually.