

Battalion Scripting Reference

Last updated 7/24/09

- complete documentation reformat (for readability)
- improved documentation for all conditions and responses
- added any missing conditions and responses currently available
- appendices built to contain useful constants

INTRODUCTION

Scripting exists in the form of **events**. Events tie together the concept of **conditions** and **responses**. Conditions are the statements which must be true before we can execute the responses. Responses are the actions performed in response to our conditions being met. On a conceptual level we are speaking in English, for example “If the game is over and the first player won, I want to pause the game and have a general congratulate the player.” The script below demonstrates this task.

Define a set of events within the `<script></script>` tags in a map file.

```
<event>
  <condition type="endGame">
    <player value="1"/>
    <victory value="true"/>
  </condition>

  <response type="disableControllers">
    <disable value="true"/>
  </response>

  <response type="chat">
    <sender value="A General"/>
    <message value="You did it! You won!"/>
    <avatar value="mullen"/>
    <color value="FFFF0000"/>
  </response>

  <response type="disableControllers">
    <disable value="false"/>
  </response>
</event>
```

This event defines a single condition, the EndGame condition. This condition has some special requirements. It will only work if player 1 has achieved victory at the end of the game. Once this condition is met, all the responses will be executed. The first response, DisableControllers, turns off all players' input into the game. This effectively pauses the game. It then displays a Chat response which will pop up a message from A General. Once the player is done reading that message, another DisableControllers response is used to return control to the players.

You can have as many events as you like in any map. You can have as many responses as you like in an event. There is only a single "condition" in any one event. If you require multiple conditions, you can combine them according to the rules of boolean logic. These different logical structures are described at the start of the chapter on conditions.

KEY

- **BOOLEAN** indicates acceptable values of "true" or "false"
- **TILEX, TILEY** indicates tile map coordinates, starting with 0,0 as the top left tile
- **STRING** is a string. Examples would be "reachedBunker" or "Bob"
- **COLOR** is an RGBA color in hex. Alpha is optional. Example 000000 or FFF0AAFF (FF=red, F0=green, AA=blue, FF=alpha)
- **PLAYER** is an integer specifying player index (starting at 1)
- **INTEGER** is a whole number. Ex 142
- **UNITTYPE** is a string. Examples "Strike Commando" "Albatross Transport". A full list is provided in Appendix A.

Responses

Chat

Use this to pop up a chat box at the bottom of the screen. The sender is the name of the person saying the message. Common convention is to put the sender value in all CAPS. Avatar is a string lookup for an avatar image. For a listing of these lookups use Appendix A. Color is a thin bar between the avatar and the message, used to show the sender's alignment with other speakers. A general for the Northern Federation army will use red, for example.

There are some special characters you can put into the message and the game will automatically replace those characters with extra information. Placing the caret (^) before a player number will print out the player's name. ^1 is player 1's name, ^2 is player 2's name, etc. Placing the ampersand (&) before a player number will print out the player's team. &1 might print out "Northern Federation", for example.

```
<response type="chat">
  <sender value="STRING" />
  <message value="STRING"/>
  <avatar value="AVATAR_REFERENCE"/>
  <color value="COLOR"/>
</response>
```

Possible values for avatar_reference are listed in Appendix A.

CreateTransportedUnit

Places a new unit, transportType, at the tile with coordinates x,y. The unit is already placed inside a transport vehicle, specified by transporterType. Player is the owner of the unit. This doesn't do any checks to make sure that the transporterType is valid, and it doesn't check to see if the transporterType can actually exist at the given tile. This means you can technically put a Leviathan on land, but it will do nothing but take up space and reflect poorly on your worth as a map maker.

```
<response type="createTransportedUnit">
  <x value="TILEX" />
  <y value="TILEY" />
  <transportType value="UNITTYPE" />
  <transporterType value="UNITTYPE" />
  <player value="PLAYER" />
</response>
```

CreateUnit

Places a new unit, type, at the tile with coordinates x,y. The "resources" parameter is optional. It's only useful for creating War Machines with a specific amount of resources. If you omit it, the War Machine will be created with the default quantity of resources. If you specify a resource quantity for a non-warmachine unit, it will be ignored. You can specify whether the unit will fade in or just appear automatically. Fade-ins are good for mid-game events since appearing automatically can be jarring to the player. If parallel is set to true, all the createUnitResponses in this event will fade in together. Otherwise set parallel to false and they will fade in in sequence, one by one.

```
<response type="createUnit">
  <x value="TILEX" />
```

```

    <y value="TILEY" />
    <type value="UNITTYPE" />
    <resources value="INTEGER" />
    <player value="PLAYER" />
    <fadeIn value="BOOLEAN" />
    <parallel value="BOOLEAN" />
</response>

```

DamageUnit

This damages a unit at the tile with coordinates x,y. Units have HP, you can see it in the recon panel when playing the game. The amount of damage is dealt directly to this HP value.

```

<response type="damageUnit">
    <x value="TILEX" />
    <y value="TILEY" />
    <amount value="INTEGER" />
</response>

```

DecrementCounterVariable

Decreases a counter with the given name by one. If this is the first time you're naming the counter, it starts at zero (and would thusly get decremented to -1.) Use this in conjunction with IncrementCounterVariable responses and CounterVariable conditions to set up custom conditions like "when the player kills 15 scorpions."

```

<response type="decrementCounterVariable">
    <counter value="STRING"/>
</response>

```

DestroyUnit

Kills the unit at the tile with coordinates x,y.

```

<response type="destroyUnit">
    <x value="TILEX" />
    <y value="TILEY" />
</response>

```

DisableAllUnits

Makes it impossible for the given player to build units from any building or unit that lets you make new units. This will not prevent the player from getting units via conditions and responses. Set disable to false to allow the player to build units again. Use this together with the EnableUnit response and you can limit what units are buildable on your map.

```

<response type="disableAllUnits">
    <disable value="BOOLEAN" />
    <player value="PLAYER" />
</response>

```

DisableControllers

Turns off all input to the game window from all players. It is usually a good idea to do this during a

long string of responses that you want to fire. Some examples are demonstrations on the map during a tutorial or conversations using the Chat response. Send another DisableController with disable set to false when you want to re-enable input.

```
<response type="disableControllers">
  <disable value="BOOLEAN"/>
</response>
```

EnableUnit

Allows a player to build the specified unit from any building or unit that lets you make new units. This does not affect units acquired from other conditions and responses. Depending on whether you set enable to true or false, you can toggle the units availability. Use this together with the DisableAllUnits response and you can limit what units are buildable on your map.

```
<response type="enableUnit">
  <enable value="BOOLEAN" />
  <player value="PLAYER" />
  <unit value="UNITTYPE" />
</response>
```

Highlight

Draws indicators around a tile at the given coordinates x,y. This is useful for pointing out units that are important during a tutorial or briefing, or terrain tiles that are important to the level. Make another Highlight response with show set to false to hide the indicator when you are done.

```
<response type="highlight">
  <x value="TILEX" />
  <y value="TILEY" />
  <show value="BOOLEAN" />
</response>
```

IncrementCounterVariable

Increases a counter with the given name by one. If this is the first time you're naming the counter, it starts at zero (and would thusly get incremented to 1.) Use this in conjunction with DecrementCounterVariable responses and CounterVariable conditions to set up custom conditions like "when the player kills 15 scorpions."

```
<response type="incrementCounterVariable">
  <counter value="STRING" />
</response>
```

NameUnit

Gets the unit at the given tile coordinates x,y and assigns a unique name to it. Multiple units can share these unique names. This lets you make certain units special without affecting the entire unit type. One scorpion can be named so that if it dies you will find out via a KillUnit condition, while the other scorpions remain unaffected. Naming a unit will also report MoveUnit conditions for the named unit. If a unit is not named, these conditions are still fired but they will use the unit's default name. See Appendix A for default unit names.

```
<response type="nameUnit">
```

```
    <x value="TILEX" />
    <y value="TILEY" />
    <name value="STRING" />
</response>
```

RemoveUnit

Removes a unit at the given tile coordinates x,y from the game board. You can use fadeOut to determine whether the unit will be simply removed or if it will fade out before it is removed. This is not the same as DestroyUnit because the unit is not killed. Therefore, no KillUnit conditions are satisfied. You can still cause a player to lose if you remove the player's final unit.

```
<response type="removeUnit">
  <x value="TILEX" />
  <y value="TILEY" />
  <fadeOut value="BOOLEAN" />
</response>
```

Resources

Awards the specified player an amount of resources. This is not guaranteed to work as expected if you give negative values for amount.

```
<response type="resources">
  <player value="PLAYER" />
  <amount value="INTEGER" />
</response>
```

Scroll

Moves the camera to focus on the tile with given coordinates x,y.

```
<response type="scroll">
  <x value="TILEX"/>
  <y value="TILEY"/>
</response>
```

SendAction

Performs certain automated tasks. An **action** is usually done by the player, like when they decide to move a unit, attack another unit, build a new unit, etc. Some actions are not able to be sent with SendAction. See Appendix B for a full list of actions and their corresponding xml.

```
<response type="sendAction">
  ACTION_XML
</response>
```

See Appendix B for examples of ACTION_XML.

SendTrigger

Lets you custom control when a condition fires. A **trigger** is responsible for changing the state of a condition. For example, whenever a unit dies, a KillUnit trigger will fire, telling the KillUnit condition that it has been met. Appendix C has a full list of available triggers and their corresponding xml.

```
<response type="sendTrigger">
  TRIGGER_XML
</response>
```

See Appendix C for examples of TRIGGER_XML

SetBooleanVariable

Changes the state of a boolean variable. If the variable does not exist yet, it will be created. Use this together with the BooleanVariable condition to trigger your own custom conditions.

```
<response type="setBooleanVariable">
  <boolean value="STRING" />
  <value value="BOOLEAN"/>
</response>
```

StructureResources

Creates a pool of resources on the structure at the tile given by the coordinates x,y. In old school maps that have buildings in them, buildings are able to produce resources. Use this response to set the number of resources available to the owner of the building. As long as they own the building, it will produce up to a limit of resources each turn until the pool is depleted.

```
<response type="structureResources">
  <x value="TILEX" />
  <y value="TILEY" />
  <amount value="INTEGER" />
</response>
```

Trace

Internal use only.

```
<response type="trace">
  <message value="STRING" />
</response>
```

UnitDisable

This will disable the unit at the given tile coordinates x,y. Disabling in this sense means that the unit will be greyed out and the controlling player can't command the unit for the remainder of their turn. You can also set disable to false to re-enable a unit that reached the tile. Do not confuse this with DisableAllUnits or EnableUnits, which modify the availability of units that can be built.

```
<response type="unitDisable">
  <x value="TILEX" />
  <y value="TILEY" />
  <disable value="BOOLEAN" />
</response>
```

CONDITIONS

LogicalAND

CONDITIONS is a subset of conditions. If they all resolve to true, then this condition resolves to true as well.

```
<condition type="and">
  CONDITIONS
</condition>
```

LogicalOR

CONDITIONS is a subset of conditions. If any one of the sub conditions resolves to true, this condition resolves to true as well.

```
<condition type="or">
  CONDITIONS
</condition>
```

LogicalNOT

CONDITION is a condition. This resolves to the opposite of the subcondition.

```
<condition type="not">
  CONDITION
</condition>
```

LogicalXOR

CONDITIONS is a subset of conditions. True if, and only if, exactly one of the sub conditions is true.

```
<condition type="xor">
  CONDITIONS
</condition>
```

BooleanVariable

This condition will fire once the boolean variable specified by boolean equals the value. You can modify these boolean variables using the SetBooleanVariable response.

```
<condition type="booleanVariable">
  <boolean value="STRING" />
  <value value="BOOLEAN" />
</condition>
```

BuildUnit

This fires when a unit is built. Specify the unit with unitName. The rest of the parameters are optional. You can let this condition only fire when a certain player builds a unit, otherwise set player to "". Likewise with x and y, you can have this condition fire only when a unit is built on a certain tile. If you don't care, leave both x and y as "".

```
<condition type="buildUnit">
  <unitName value="UNITTYPE" />
```

```

    <player value="PLAYER" />
    <x value="TILEX" />
    <y value="TILEY" />
</condition>

```

CaptureTerrain

This fires whenever a terrain is captured. All parameters are optional. You can further restrict when this condition fires by giving a type of terrain, the previous owner of the terrain, or the new owner. You should set any of these values that you do not provide to "". A valid list of terrainName values is provided in Appendix A.

```

<condition type="captureTerrain">
    <terrainName value="STRING" />
    <oldPlayer value="PLAYER" />
    <player value="PLAYER" />
</condition>

```

See Appendix A for valid terrainName values.

CounterVariable

This condition will fire once the counter variable specified by counter equals the value. You can modify these counter variables using the IncrementCounterVariable and DecrementCounterVariable responses. Set defaultValue if you want the counter to start at something other than zero.

```

<condition type="counterVariable">
    <counter value="STRING" />
    <value value="INTEGER" />
    <defaultValue value="INTEGER" />
</condition>

```

EndGame

Fired when the game ends. By default this will fire when someone wins but you can use the two optional parameters to narrow this down to a particular player and a particular victory state (true for win and false for lose.)

```

<condition type="endGame">
    <player value="PLAYER" />
    <victory value="BOOLEAN" />
</condition>

```

KillUnit

Fired whenever a unit is killed. The rest of the parameters are optional and let you further customize when this condition is fired. You can restrict it to certain unit types, or specialized units that you named with the NameUnit response. You can have this fire only when a certain player loses units, or when a unit is lost on a specific tile. Set any optional parameters that you do not use to "".

```

<condition type="killUnit">
    <unitName value="UNITTYPE" />
    <player value="PLAYER" />
    <x value="TILEX" />

```

```
        <y value="TILEY" />
    </condition>
```

MoveUnit

This is exactly like KillUnit condition, except when a unit moves instead of when it is killed. Read KillUnit for more information on what these optional parameters do.

```
<condition type="moveUnit">
    <unitName value="UNITTYPE" />
    <player value="PLAYER" />
    <x value="TILEX" />
    <y value="TILEY" />
</condition>
```

ResourcesAcquired

Anytime a player gets new resources, this will fire. Use the optional parameters to filter the condition further. You can filter on a specific player acquiring the resources. Setting resources means that the condition won't fire unless the resources acquired is at least equal to or greater than what value you provide. Everytime resources are acquired an internal counter keeps track of how many resources the player has gained. You can set the count parameter if you want this internal counter to fire the ResourcesAcquired condition once it is at least equal to or greater than the value you provide.

```
<condition type="resourcesAcquired">
    <resources value="INTEGER" />
    <player value="PLAYER" />
    <count value="INTEGER" />
</condition>
```

StartGame

Fires when the game starts. StartGame is a very important condition as it allows you to prepare any special scripting you'd like the game to have, before the players get control of the game.

```
<condition type="startGame">
</condition>
```

StartTurn

This fires at the start of every turn. You can use the optional parameters to filter down to the start of a specific player's turn, or a specific turn count. This is useful for implementing turn time limits on maps, like defending something for 15 turns. Remember that without a player parameter, each player's turn will fire this. This means that it isn't an accurate round counter (a **round** is each player taking one turn) without the player parameter.

```
<condition type="startTurn">
    <player value="PLAYER" />
    <round value="INTEGER" />
</condition>
```

TimePassedCondition

All time is provided in milliseconds (1000 ms is 1 second). This condition will fire once a certain

amount of time has passed. Time is the amount of time that you would like to pass before the condition fires, like 10,000 ms for 10 seconds. Timer lets you start the clock at something other than zero. If you don't want to start it at anything other than zero, you can set timer to "".

```
<condition type="timePassed">  
  <time value="INTEGER" />  
  <timer value="INTEGER" />  
</condition>
```

Appendix A – Valid values for variables

Valid UNITTYPE values

Ground Units

- Strike Commando
- Heavy Commando
- Flak Tank
- Scorpion Tank
- Mortar Truck
- Rocket Truck
- Annihilator Tank
- Turret
- Stealth Tank
- Warmachine
- Jammer Truck
- Blockade
- Lancer Tank
- Spider Tank

Air Units

- Albatross Transport
- Raptor Fighter
- Condor Bomber

Sea Units

- Leviathan Barge
- Hunter Support
- Corvette Fighter
- U-Boat
- Battlecruiser
- Intrepid

Valid TerrainName values

Terrain

- Plains
- Sea
- Road
- Forest
- Mountain
- Desert
- Reef
- Shore
- Bridge
- Hill
- Canyon
- Wasteland
- Volcano
- Archipelago
- Rock Formation
- High Bridge
- Enriched Ore Deposit
- Depleted Ore Deposit
- Ore Deposit

Buildings

- Command Center
- Air Control
- Ground Control
- Sea Control
- Factory
- Oil Refinery
- Advanced Oil Refinery
- Oil Rig

Valid AVATAR_REFERENCE Values

Episode 1 (Warning: contains spoilers)

- durand
- mullen
- pearl
- tucker
- argent
- general
- evil_durand
- Akadian_2
- Akadian_3
- evil_durand_dark
- Akadian_5

Appendix B – Valid ACTION_XML Values

Surrender

Causes the given player to surrender.

```
<Action stateCode="646" isFromNetwork="true" isNetworked="true">  
  <SurrenderAction player="1"/>  
</Action>
```

Appendix C – Valid TRIGGER_XML Values

There are none!