IGN Functions List and Documentation

Igneous01

Contents

IGN_fnc_createEvent

Creates an event object, allowing registration of handlers and raising
Does not broadcast in multiplayer

Parameters:
args (Array) (Optional) - Argument types that the event will send to all handlers
default value - [] (no arguments)
name (String) (Optional) - The name of the object (this will set the vehicle's variable name)
default value - "" (no name)

Returns:
The created event object, with these getters in its namespace:
"name" - the name of the event (Argument passed)
"handlers" - an array containing all handlers currently registered to the event
"arg_types" - an array containing the argument types
"raised" - a bool value determining if the event is currently being raised

Example:

```
myEvent = call IGN_fnc_createEvent;          // no args, no vehicle var name

myEvent = [[objNull, []]] call IGN_fnc_createEvent;
// will send arguments of type object (_this select 0) and array (_this select 1)

[objNull, "myEvent"] call IGN_fnc_createEvent;
// myEvent can be referenced, will send argument of type object (_this)

[[objNull], []], "myEvent"] call IGN_fnc_createEvent;
// vehicle var name will be myEvent - can now reference myEvent in code
```

IGN_fnc_createEventServer

Creates an event object on the server only, allowing registration of handlers and raising
Broadcasts vehicleVarName (if exists) and setVariables in object space

Parameters:
args (Array) (Optional) - Argument types that the event will send to all handlers
default value - [] (no arguments)
name (String) (Optional) - The name of the object (this will set the vehicle's variable name)
- Will be broadcasted in MP
default value - "" (no name)

Returns:
The created event object, with these getters in its namespace:
"name" - the name of the event (Argument passed)
"handlers" - an array containing all handlers currently registered to the event
"arg_types" - an array containing the argument types
"raised" - a bool value determining if the event is currently being raised

Example:

```
myEvent = call IGN_fnc_createEventServer;          // no args, no vehicle var name

myEvent = [[objNull, []]] call IGN_fnc_createEventServer;
// will send arguments of type object (_this select 0) and array (_this select 1)

[[objNull], []], "myEvent"] call IGN_fnc_createEventServer; // vehicle var name will be myEvent
```

IGN_createEventClient

Creates an event object on the client(s) only, allowing registration of handlers and raising
Does not broadcast the object in multiplayer

Parameters:
args (Array) (Optional) - Argument types that the event will send to all handlers
    default value - [] (no arguments)
name (String) (Optional) - The name of the object (this will set the vehicle's variable name)
    default value - "" (no name)

Returns:
The created event object, with these getters in its namespace:
"name" - the name of the event (Argument passed)
"handlers" - an array containing all handlers currently registered to the event
"arg_types" - an array containing the argument types
"raised" - a bool value determining if the event is currently being raised

Example:

```
myEvent = call IGN_fnc_createEventClient; // no args, no vehicle var name

myEvent = [[objNull, []]] call IGN_fnc_createEventClient;
// will send arguments of type object (_this select 0) and array (_this select 1)

[[objNull], []], "myEvent"] call IGN_fnc_createEventClient; // vehicle var name will be myEvent
```

IGN_fnc_createEventLocal

Creates a local event object using CreateVehicleLocal (not broadcasted in MP), allowing registration of handlers and raising
Does not broadcast the object in multiplayer

Parameters:
args (Array) (Optional) - Argument types that the event will send to all handlers
default value - [] (no arguments)
name (String) (Optional) - The name of the object (this will set the vehicle's variable name)
default value - "" (no name)

Returns:
The created event object, with these getters in its namespace:
"name" - the name of the event (Argument passed)
"handlers" - an array containing all handlers currently registered to the event
"arg_types" - an array containing the argument types
"raised" - a bool value determining if the event is currently being raised

Example:

```
myEvent = call IGN_fnc_createEventLocal; // no args, no vehicle var name

myEvent = [[objNull, []]] call IGN_fnc_createEventLocal;
// will send arguments of type object (_this select 0) and array (_this select 1)

[[objNull], []], "myEvent"] call IGN_fnc_createEventLocal; // vehicle var name will be myEvent
```

IGN_fnc_deleteEvent

Deletes an existing event object
If an invalid event object is passed, it will not be deleted

Parameters:
event (object) - the event to be deleted

Returns:
nothing

Example:

```
myEvent call IGN_fnc_deleteEvent;
[myEvent] call IGN_fnc_deleteEvent;
```

IGN_fnc_raiseEvent

Raises an existing event, calling all handlers in sequence with specified arguments
When an event object is raised, it will set its namespace attribute "raised" to true, until all handlers calls are finished
When all handlers have received the event, the attribute "raised" will be set to false

Parameters:
event (object) - the event to be raised
arguments (anything) (optional) - the arguments to be passed to each handler
default value - [] (no arguments will be passed)

Returns:
nothing

Example

```
[myEvent] call IGN_fnc_raiseEvent;  // raises myEvent, no arguments are passed to handlers

myEvent call IGN_fnc_raiseEvent;    // same as above, but without array brackets

[myEvent, [myString, myTank]] call IGN_fnc_raiseEvent;
// raises myEvent, passes myString as (_this select 0), myTank as (_this select 1) to all handlers

[myEvent, myTank] call IGN_fnc_raiseEvent;
// raises myEvent, passes myTank as _this to all handlers
```

ASYNCHRONOUS/NON-SCHEDULED/THREADED calls to an event

This is particularly useful if you have spawn'ed threads that may raise the event and want to synchronize calls

Example of Unsafe non-scheduled calls to an event:

```
// thread 1 only tracks player death
_thread_1 = spawn {
	waitUntil {!alive player};
	[eventUnitDead, player] call IGN_fnc_raiseEvent;
	// this can be called at the same time as the other one!
};

// thread 2 tracks any dead in enemyUnits
_thread_2 = spawn {
	private ["_grpAliveCount"];
	_grpAliveCount = count enemyUnits;
	while {_grpAliveCount > 0} do
	{
		{
			if (!alive _x) then
			{
				_grpAliveCount = _grpAliveCount - 1;
				[eventUnitDead, _x] call IGN_fnc_raiseEvent;
				// this can be called at the same time as the other one!
			};
		} foreach enemyUnits;
```

```
	};
};

// Potential for thread 1 and thread 2 to raise eventUnitDead at the same time! not safe!
```

Example of safer non-scheduled calls to an event

```
// thread 1 only tracks player death
_thread_1 = spawn {
	waitUntil {!alive player};
	waitUntil {!(eventUnitDead getVariable "raised")}; // block thread if event is busy
	[eventUnitDead, player] call IGN_fnc_raiseEvent;
};

// thread 2 tracks any dead in enemyUnits
_thread_2 = spawn {
	private ["_grpAliveCount"];
	_grpAliveCount = count enemyUnits;
	while {_grpAliveCount > 0} do
	{
		{
			if (!alive _x) then
			{
				_grpAliveCount = _grpAliveCount – 1;
				// block thread if event is busy
				waitUntil {!(eventUnitDead getVariable "raised")};
				[eventUnitDead, _x] call IGN_fnc_raiseEvent;
			};
		} foreach enemyUnits;
	};
};
```

IGN_fnc_addEventHandler

Adds a handler to the specified event, handlerID is returned

Parameters:
event (object) - the event
handle (code) - event handler code

Returns:
handlerID (index of handler in this event)

Example:

```
handlerID = [myEvent, {hint format ["%1", _this];}] call IGN_fnc_addEventHandler;
// everytime myEvent is raised, a hint will display the argument passed to the handler (assuming myEvent sends 1 argument or array)

// Assuming myEvent was raised as such : [myEvent, ["weapon_class_tank_cannon", someTank]] call fnc_RaiseEvent;

handlerID = [myEvent,
{
	private ["_weaponName", "_tank"];
	_weaponName = _this select 0;		// raise of myEvent sends weapon string as arg 0
	_tank = _this select 1;			// tank object is sent as arg 1
	//...
}] call IGN_fnc_addEventHandler;
```

IGN_fnc_deleteEventHandler

removes a handler from the specified event

Parameters:
event (object) - the event
handleID (number) - event handler id

Returns:
nothing

Example:

```
[myEvent, handlerID] call IGN_fnc_deleteEventHandler;	// removes handlerID from myEvent
```

IGN_fnc_getEventHandlers

returns the array of all event handlers registered to the current event

Parameters:
event (object) - the event

Returns:
array of code

Example:

```
allHandlers = myEvent call IGN_fnc_getEventHandlers;
```

IGN_fnc_getEventHandler

returns the specified handler at the specified index

Parameters:
event (object) - the event
id (number) - index of handler

Returns:
code

Example:

```
handler = [myEvent, 0] call IGN_fnc_getEventHandler;      // returns first registered handler
```

IGN_fnc_getEventHandlerCount

returns the size of the handler array

Parameters:
event (object) - the event

Returns:
size (number)

Example:

```
numHandles = myEvent call IGN_fnc_getEventHandlerCount;
```

IGN_fnc_createFunctor

Creates a new functor object
arguments passed cannot be temporary values - they must be references to existing objects/tasks/groups/etc...

Parameters:
function (code) - the code the functor will call
arguments (*anything) - the arguments the functor will pass to the function (Must be References!!)

Returns:
Functor object, with these attributes
"**IGN_FUNCTOR_TYPE**" - type identifier
"**function**" - code
"**arguments**" - array of pointers to original arguments (if any)

Example:

```
myFunctor = [
{
	private ["_unit"];
	_unit = _this;
	_unit setUnitPos "MIDDLE";
}, _myUnit] call IGN_fnc_createFunctor;
// notice that _myUnit is local - this means you can call myFunctor outside of
// the script and it will still reference _myUnit
```

In the above example, myFunctor passes one argument (_myUnit) and will return nothing

Example:

```
myFunctor = [
{
	private ["_units"];
	_units = _this;
	{ _x setUnitPos "MIDDLE"; } foreach _units;
	true;
}, [_myUnit1, myUnit2]] call IGN_fnc_createFunctor;

// This time, myFunctor passes two arguments (local _myUnit and global myUnit2) and will return
// true upon completion
```

IGN_fnc_deleteFunctor

Deletes an existing functor, as well as all of its pointers

Parameters:
functor (object) - the functor to be deleted

Returns:
nothing

Example:

```
myFunctor call IGN_fnc_deleteFunctor;
[myFunctor] call IGN_fnc_deleteFunctor;
```

IGN_fnc_callFunctor

Calls the functors code, passing in the stored arguments
Arguments are dereferenced upon call

Parameters:
functor (object) - the Functor to call

Returns:
anything (function) - If the specified code returns a value(s), this will return that value(s)

Example:

```
myResult = myFunctor call IGN_fnc_callFunctor;
[myResult] = myFunctor call IGN_fnc_callFunctor;
```