VBS User Interface Event Handlers

This page is obsolete. For latest VBS User Interface Event Handlers, see UI Event Handlers in the VBS3 Scripting Manual.

Overview

User Interface Event Handlers constantly monitor displays and controls, and execute custom code when a particular UI events has being triggered.

Defining events

Event Handlers can be assigned in two ways: via class property definitions (in `description.ext` or an addon `config.cpp`) or via scripting commands.

A control must be ctrlEnable, in order for any event to be able to fire.

Independently of how the event was defined, it can be removed via the following script commands: ctrlRemoveEventHandler & ctrlRemoveAllEventHandlers and displayRemoveEventHandler & displayRemoveAllEventHandlers.

Event properties

Event parameters

Event handlers receive parameters via the _**this** variable, which is an array containing event-specific information (e.g. the control clicked and the mouse position - see Reference List). The content of this variable can then be used in the command string that is defined for this event.
Example: Event sends a chat message if the right mouse button is clicked over the control:

```
onMouseButtonDown = "if ((_this select 1)==2) then {player sidechat 'Right mouse button clicked'}";
```

Command string

The string that is passed to the event handler can contain one or more commands, which are separated by semicolons.
Local variables can be used in the command string.
Example: Mouse double-click event stores the passed text control in variable '_ctrl', saves the original content of it into 'tst_var', and then clears the control:

```
onMouseButtonDblClick = "_ctrl=_this select 0; tst_var=ctrlText _ctrl; _ctrl ctrlSetText ''";
```

Return values

Once the commands in the passed string have been executed, the event handler returns a Boolean to the engine (the result of the last command in the string). If the returned value is true (i.e. "event completed"), then no further processing of the event will happen; if it is false then normal event processing continues.
This is really only of interest in two situations: Key-press events ("onKeyDown") and multiple (stacked) handlers for the same type of event.

- If a *key-press event* returns true, then this will prevent the engine from interpreting the <Esc> key, and the dialog cannot be closed via that method anymore (and the designer will have to close it by some other way). In order to prevent that the command string should end with a false flag: onKeyDown = "doSomething; false".
- If *multiple handlers* are defined for the same event type (via ctrlAddEventHandler or displayAddEventHandler) and the command string returns true , then only the first event will be executed, and any further definitions are ignored. This can be circumvented (just like in the key-press situation) by ending the string with an explicit false:
  _ctrl ctrlAddEventHandler ["MouseButtonDown","systemChat 'EH #1'; false"];
  _ctrl ctrlAddEventHandler ["MouseButtonDown","systemChat 'EH #2'; false"];

The 'onBPS' event is a special situation, where the return value is a String. This string contains the command(s) that are executed when a branch point is *lo aded*. (And as a string is interpreted as a false return value, multiple events can thus be defined.)

Class defined events

Events can be defined in the Dialog (display) or Control classes (in `config.cpp` or `description.ext`). The event property value (string) is executed as a line of code. An example line (this would be put within a control or dialog class):

```
onMouseDown = "hint str _this";
```

Example:

```
class DlgInput {
  idd = 20000;
  movingEnable = true;

  class controls {
    class EDIT1 : RscEdit {
      idc = 20000;
      x = .2; y = .2; w = .2; h = .2;
      onMouseButtonDblClick = "(_this select 0) ctrlSetText ''";
    };
  };
};
```

Script defined events

Events can also be defined via scripting commands: ctrlSetEventHandler & ctrlAddEventHandler, and displaySetEventHandler & displayAddEventHandler.

Important: When using event names with a command, the prefix "on" must be omitted. (e.g. '**ButtonDown'** instead of '**onButtonDown'**)

```
(findDisplay 46) displaySetEventHandler ["keyDown", "_this execVM 'eventScript.sqf'"];
```

Reference List

Not all events can be triggered on all types of dialog types. Some will only work with displays, others will only work with specific control types. The possible types are listed below, and the right column ("Scope") shows which controls the event can be used with.
Most user input events (e.g. mouse clicked, keyboard pressed) will *not* work with HUD-like dialogs (i.e. those defined as RscTitles), since the dialog itself is not interactive.

- **D**: VBS Displays
- **D2**: VBS Displays (*V2.0+*)
- **A**: Dialogs_ActiveText
- **AT**: Dialogs Tree (*V3.4+*)
- **B**: Dialogs Button
- **Ch**: Dialogs Checkboxes
- **C**: Combobox
- **CM**: *Content Menu*
- **G**: Dialogs Group
- **H**: Dialogs HTML
- **I**: Images
- **L**: Dialogs Listbox
- **M**: Dialogs Map
- **O**: *Object*
- **PV**: Dialogs PlanView (*V3.6+*)
- **S**: Dialogs Static
- **Sb**: Dialogs Statebox (*V3.4+*)
- **Sl**: Dialogs Slider
- **Str**: Dialogs StructuredText
- **T**: Dialogs TextBox
- **Tr**: *Tree*
- **To**: Dialogs Toolbox

(Entries in italics have not been verified.)

Event	Fired	Notes and Parameters	Scope
onLoad	Display and all controls are created. Note that findDisplay will not be able to find the newly created display until after the onLoad event has finished processing. If you require access to the display in your onLoad event handler, use the handle passed in the **_this** parameter.	Returns the display.	D

onUnload	Display is closed, but no controls are destroyed yet.	Returns the display and exit code. The exit code is 2 if the dialog was closed by pressing the <Esc> key, otherwise it's the number that was passed via the closeDialog command.	D
onChildDestroyed	Child display is closed.	Returns the display, which child display was closed and exit code.	*D*
onBPS	A multiplayer session is either saved or loaded (*BPS==Branch Point Save*), while the current user has a dialog opened.	Returns the display for save events, and nothing for load events. Any command(s) defined in the passed string will be executed when a branchpoint (BP) is *saved*; the command(s) that should be executed when a BP is *loaded* have to be enclosed in quotation marks (i. e. passed as a string within the command string), and be the last (preceded by a semicolon), or only, element: "saveAction; 'loadAction'". The *saveAction* is executed when a branchpoint is saved, the (optional) *loadAction* is passed as a string, and executed when a branchpoint is loaded. (e.g. onBPS = "TST_COUNT=TST_COUNT+1; 'TST_COUNT=TST_COUNT-1'")	*D* (V3.4+)
onRearrange	The aspect ratio has been changed (e.g. via the video options).	Returns the display.	*D* (V3.4+)
onTimer	After the time defined in a control's "timer" property has passed, since the control has been opened.	Returns control.	*Any dialog control* (V3.4+)
onMouseEnter	The mouse pointer enters the control area.	Returns control.	A, B, T, Sb, To
onMouseExit	The mouse pointer exits the control area.	Returns control.	A, B, T, Sb, To
onSetFocus	Input focus is on control. It now begins to accept keyboard input.	Returns control.	A, AT, B, C, G, H, L, M, Sb, Sl, SV, T, To
onKillFocus	Input focus is no longer on control. It no longer accepts keyboard input. A control only loses focus if another one receives it - just leaving the control bounding box does not kill focus.	Returns control.	A, AT, B, C, G, H, L, M, Sb, Sl, SV, T, To
onKeyDown	Pressing any keyboard key. Fired before the onKeyUp event.	Returns the control, the DIK KeyCodes and the state of Shift, Ctrl and Alt. Be aware that if this EH is attached to a display, and its function doesn't return false, the <Esc> key will become non-functional, and you will not be able to close the dialog. This can be used to make the dialog closure dependent on certain condition, e.g. onKeyDown = "if (somecondition || (_this select 1)! =1) then {false} else {true}"	A, AT, B, C, D, G, H, L, Ln, M, Sb, Sl, SV, T, To
onKeyUp	Releasing any keyboard key. Fired after the onKeyDown event.	Returns the control, the DIK KeyCodes and the state of Shift, Ctrl and Alt.	A, AT, B, C, D, G, H, L, M, Sb, Sl, SV, T, To
onChar	When some readable characters is recognized.	Returns the control and the char code.	A, AT, B, C, D, G, H, L, M, Sb, Sl, SV, T, To
onIMEChar	When IME character is recognized (used in Korean and other eastern languages).	Returns the control and the char code.	*Control*
onIMEComposition	When partial IME character is recognized (used in Korean and other eastern languages).	Returns the control and the char code.	*Control*
onJoystickButton	Pressing and releasing any joystick button.	Returns the control and the the pressed button.	*Control*
onMouseButtonDown	Pressing a mouse button. Followed by the onMouseButtonUp event.	Returns the control, the pressed button (0:left, 1:right, etc.), the x and y coordinates and the state of Shift, Ctrl and Alt. For some displays (e.g. 46: editor) the value for x & y is always [.5,.5]. If a display is overlaid by a dialog, the display EH will *still* fire, even when the dialog is clicked!	A, AT, B, C, D2, G, H, L, M, Sb, Sl, SV, T, To
onMouseButtonUp	Releasing a mouse button. Follows the onMouseButtonDown event.	Returns the control, the pressed button (0:left, 1:right, etc.), the x and y coordinates and the state of Shift, Ctrl and Alt. For some displays (e.g. 46: editor) the value for x & y is always [.5,.5]. If a display is overlaid by a dialog, the display EH will *still* fire, even when the dialog is clicked!	A, AT, B, C, D2, G, H, L, M, Sb, Sl, SV, T, To
onMouseButtonClick	Pressing and releasing a mouse button.	Returns the control, the pressed button (0:left, 1:right, etc.), the x and y coordinates, and the state of Shift, Ctrl and Alt.	A, AT, B, C, G, H, L, M, Sl, SV, T, To

onMouseButtonDblClick	Pressing and releasing a mouse button twice within very short time.	Returns the control, the pressed button (0:left, 1:right, etc.), the x and y coordinates and the state of Shift, Ctrl and Alt.	A, AT, B, C, G, H, L, M, Sl, SV, T, To
onMouseMoving	Fires continuously while moving the mouse with a certain interval.	Returns the control, the x and y coordinates, and a Boolean indicating whether currently over a compatible control. (V2.00+: For displays or non-compatible controls the display and mouse speed x,y is returned.)	A, AT, B, C, D2, G, H, L, M, Sb, Sl, SV, T, To
onMouseHolding	Fires continuously while mouse is not moving with a certain interval.	Returns the control, the x and y coordinates, and a Boolean indicating whether currently over a compatible control. (V2.00+: For displays or non-compatible controls the display and relative mouse speed x,y is returned.)	A, AT, B, C, D2, G, H, L, M, Sb, Sl, SV, T, To
onMouseZChanged	Fires when mouse wheel position is changed.	Returns the control and the number of lines scrolled (depends on scroll-speed and system settings). Values are positive for forward scrolls or negative for backward scrolls.	A, AT, B, C, D2, G, H, L, M, Sb, Sl, SV, T, To
onCanDestroy	Ask this control if dialog can be closed (used for validation of contained data).	Returns the control and exit code (0=closed via closeDialog, 2=closed via <Esc>).	*?*
onDestroy	Destroying control	Returns the control and exit code.	*?*
onButtonClick	The attached button action is performed.	Returns control.	A, B
onButtonDown	The left mouse button is pressed over the button area or a key on the keyboard is pressed.	Returns control.	A, B
onDraw	Fires when the map is drawn (can occur more than once per second).	Returns the map control.	M
onLBSelChanged	The selection in a listbox or combobox has changed. The left mouse button has been released and the new selection is fully made.	Returns the control and the selected element index (always only *one* -the latest one- even if multiple selections are enabled).	C, L
onLBDblClick	Double click on some row in listbox.	Returns the control and the selected element index.	L
onLBDrag	Drag & drop operation started.	Returns the control and a nested array containing [text,data,value] for all the selected elements.	L
onLBDragging	Drag & drop operation is in progress.	Returns the control currently hovered over, the current x and y coordinates, the IDC of the source control, and a nested array containing [text,data,value,index] for all the selected elements, and a -1 as the last value (unknown meaning). In 2D listboxes the element index is calculated by counting every cell in every line (e.g. the 1st cell in line 3 -each containing 2 cells- would return 4).	L
onLBDrop	Drag & drop operation finished.	Returns the control item was dropped onto, the current x and y coordinates, the IDC of the source control, and a nested array containing [text,data,value,index] for all the selected elements, and the index it was dropped into (or -1 if dropped onto a non-listbox or into an empty space of a listbox). In 2D listboxes the element index is calculated by counting every cell in every line (e.g. the 1st cell in line 3 -each containing 2 cells- would return 4).	A, B, C, H, L, M, Sb, Sl, SV, T, To
onSetVisible	A control's visibility is changed via ctrlShow.	Returns the control and the visibility status (1=visible, 0=hidden).	A, AT, B, C, G, H, L, M, S, Sb, Sl, Str, SV, T, To
onAdvTreeToggle	A row was collapsed or expanded (either via a user action or a script command).	Returns the control, row number (1-based), and whether the branch is currently expanded.	AT
onAdvTreeClick	Any item in a row was clicked (down & up). If button is *still* held down after about 1 second, event will fire nevertheless.	Returns the control, row number (1-based), column number (0-based).	AT
onAdvTreeDblClick	Any item in a row was double-clicked.	Returns the control, row number (1-based), column number (0-based).	AT
onAdvTreeMouseDown	The mouse button is currently pressed over any item in a row.	Returns the control, row number (1-based), column number (0-based).	AT

onAdvTreeMouseUp	The mouse button is released over any item in a row.	Returns the control, row number (1-based), column number (0-based).	AT
onTreeLButtonDown	Pressing and releasing left mouse button on a tree.	Returns the control.	*Tr*
onTreeMouseHold	Fires continuously while mouse is not moving with a certain interval.	Returns the control.	*Tr*
onTreeMouseMove	Fires continuously while moving the mouse with a certain interval.	Returns the control.	*Tr*
onTreeMouseExit	The mouse pointer exits the tree control area	Returns the control.	*Tr*
onTreeSelChanged	Changing the selection in a tree.	Returns the control.	*Tr*
onTreeDblClick	Pressing and releasing twice on a tree.	Returns the control.	*Tr*
onTreeExpanded	The tree folder structure has been expanded.	Returns the control.	*Tr*
onTreeCollapsed	The tree folder structure has been collapsed.	Returns the control.	*Tr*
onToolBoxSelChanged	Changed the selection in a toolbox.	Returns the control and the selected element index.	To
onCheckBoxesSelChanged	Changed the selection in a checkbox.	Returns the control, the selected element index and the current state.	*Ch*
onStateBoxToggle	A statebox control was clicked.	Returns the control, current state, and previous state. If the Command string returns true, then the state will *not* be changed!	Sb
onHTMLLink	Pressing and releasing a HTML link.	Returns the control and href.	*H*
onSliderPosChanged	Changing the position of a slider.	Returns the control and the change.	Sl
onObjectMoved	Moving an object.	Returns the control and the offset on the x,y and z axes.	*O*
onMenuSelected	Some item in context menu (used now only in new mission editor) was selected.	Returns the control and the command id.	*CM*
onPlanDraw	Fires each frame the control is displayed.	Returns the control. Does **not** work if assigned via ctrlSetEventHandler!	*PV*
onplanUnitEntered	A unit entered a section covered by the plan.	Returns [vehicle, control, unit, section]. Be aware that unlike all other events, this one *doesn't* return the control as the first element!	*PV*
onPlanOnUnitLeft	A unit left a section covered by the plan.	Returns [vehicle, control, unit, section]. See note in 'onPlanUnitEntered'.	*PV*
onPlanClick	The mouse button has been pressed.	Returns the [control, x, y], with the coordinated being relative to the plan.	*PV*
onPlanUnitSelect	A unit, visible on the plan, was clicked.	Returns [control, unit, selected]. 'Selected' will toggle between true and false.	*PV*