

NeoWin Interface Designer

Advanced Controls Tutorial — Part 2

Tests 21–29 — ContextMenu · MediaPlayer · SearchBox · TreeView · ProgressSteps · CountdownTimer · ScrollingTicker · Accordion · SimpleChart

Build · Export · Test — using the NWIDCompanion plugin

Prerequisites: Part 1 (Tests 01–20) completed. Same VNW publication and Rectangle1 setup.

Revised Edition — Part 2 v2

TEST 21 Advanced Controls

ContextMenu — Right-Click Page Menu

1 Build in Designer

1. Add Context Menu from the Advanced UI palette — a small placeholder icon appears (hidden at runtime).
2. Set ID = ctx_menu
3. In the Content tab, set Items = View Details|Edit|---|Delete
4. Wire per-item events — Events tab: click ■ Add for each item.
5. Add a Label — ID = lbl_action | Caption = (empty)

Properties Panel — what to set for each control

Control ID	Property	Value
ctx_menu	ID / Items	ctx_menu / View Details Edit --- Delete
lbl_action	ID / Caption	lbl_action / (empty)

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
ctx_menu	Item_Click View Details	GoSub	OnViewDetails
ctx_menu	Item_Click Edit	GoSub	OnEdit
ctx_menu	Item_Click Delete	GoSub	OnDelete

How: In the Events tab, click ■ Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[ctx_menu.value]	Label of the clicked menu item — e.g. "Edit"
[ctx_menu.name]	ctx_menu
[ctx_menu.event]	Item_Click

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
:OnViewDetails
NWID_Set "Rectangle1" "label:lbl_action" "caption" "View Details clicked"
Return

:OnEdit
NWID_Set "Rectangle1" "label:lbl_action" "caption" "Edit clicked"
Return
```

```
:OnDelete  
NWID_Set "Rectangle1" "label:lbl_action" "caption" "Delete clicked"  
Return
```

Expected results:

- Right-click on the page background — context menu appears at cursor
- Click "Edit" — label shows "Edit clicked"
- Right-click on a TextArea or TextInput — native browser menu appears (not intercepted)
- Separator line (---) renders as a horizontal line, not clickable

Note: Only one context menu per page. The menu does not fire on text-editable controls. Use icon tokens in items for icons:
e.g. Edit: Edit|Delete: Delete

■ **PASS** ■ **FAIL** Notes: _____

TEST 22 Media Controls

MediaPlayer — Load and Control Audio/Video

1 Build in Designer

1. Add Media Player — ID = mp_player | Height ~200px
2. Wire events — Events tab: ■ Add for Play, Pause, Ended, and Time_Update.
3. Add Button — ID = btn_load | Caption = Load Track — wire Left_Click → OnLoadTrack
4. Add Label — ID = lbl_status | Caption = (empty)

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
mp_player	Play	GoSub	OnPlay
mp_player	Pause	GoSub	OnPause
mp_player	Ended	GoSub	OnEnded
mp_player	Time_Update	GoSub	OnTimeUpdate
btn_load	Left_Click	GoSub	OnLoadTrack

How: In the Events tab, click ■ Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[mp_player.filename]	Current media file path — set on Track_Changed
[mp_player.time]	Current playback position in milliseconds
[mp_player.event]	Event that fired — e.g. "Play", "Pause"

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
:OnLoadTrack
. Replace music.mp3 with a real audio file in your publication folder
NWID_Set "Rectangle1" "mediaPlayer:mp_player" "filename" "[PubDir]music.mp3"
. Track loads ready to play. Press play or use command:
. NWID_Set "Rectangle1" "mediaPlayer:mp_player" "command" "play"
Return

:OnPlay
NWID_Set "Rectangle1" "label:lbl_status" "caption" "Playing..."
Return

:OnPause
```

```
NWID_Set "Rectangle1" "label:lbl_status" "caption" "Paused"
Return

:OnEnded
NWID_Set "Rectangle1" "label:lbl_status" "caption" "Track finished"
Return

:OnTimeUpdate
NWID_Set "Rectangle1" "label:lbl_status" "caption" "Position: [mp_player.time]ms"
Return
```

Expected results:

- Click Load Track — track loads (ready to play)
- Press play — label shows "Playing..."
- During playback — label updates with position in ms
- Pause — label shows "Paused"

Note: filename loads the track without auto-starting. Use "command" with "play", "pause", "stop", "next", or "prev" to control playback.

■ **PASS** ■ **FAIL** Notes: _____

TEST 23 Advanced Controls

SearchBox — Debounced Search with Clear

1 Build in Designer

1. Add Search Box — ID = srx_find | Placeholder = Search items...
2. Wire events — ■ Add for Search_Changed and Cleared.
3. Add Label — ID = lbl_result | Caption = Type to search

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
srx_find	Search_Changed	GoSub	OnSearchChanged
srx_find	Cleared	GoSub	OnSearchCleared

How: In the Events tab, click ■ Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[srx_find.value]	Current search text — updated after debounce (~300ms)

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
:OnSearchChanged
NWID_Set "Rectangle1" "label:lbl_result" "caption" "Searching: [srx_find.value]"
Return

:OnSearchCleared
NWID_Set "Rectangle1" "label:lbl_result" "caption" "Search cleared"
Return
```

Expected results:

- Type "hello" — after a brief pause, label shows "Searching: hello"
- Click the X clear button — label shows "Search cleared"

Note: Search_Changed fires after a debounce delay (~300ms). The clear button appears when text is present.

■ **PASS** ■ **FAIL** Notes: _____

TEST 24 Advanced Controls

TreeView — Hierarchical Node Selection

1 Build in Designer

1. Add Tree View — ID = trv_files | Height ~200px
2. Set Items in the Content tab (pipe-separated, use spaces for nesting): Documents| Report.pdf| Notes.txt|Images| Photo.jpg| Logo.png
3. Expand All should be checked (default).
4. Wire event —  Add → Event: Node_Click → Action: GoSub → Value: OnNodeClick
5. Add Label — ID = lbl_node | Caption = Click a node
6. Add Button — ID = btn_load | Caption = Load Tree — wire Left_Click → OnLoadTree

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
trv_files	Node_Click	GoSub	OnNodeClick
btn_load	Left_Click	GoSub	OnLoadTree

How: In the Events tab, click  Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[trv_files.value]	Label of the clicked node — e.g. "Report.pdf"

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
:OnNodeClick
NWID_Set "Rectangle1" "label:lbl_node" "caption" "Selected: [trv_files.value]"
Return

:OnLoadTree
NWID_Set "Rectangle1" "treeView:trv_files" "items" "Projects| Project A| Design| Code| Project B|
Testing"
Return
```

Expected results:

- Tree shows Documents and Images as parent nodes with expand arrows
- Click "Report.pdf" — label shows "Selected: Report.pdf"
- Click a parent node — fires Node_Click AND toggles expand/collapse
- Click Load Tree — tree repopulates with new hierarchy

Note: Nesting uses leading spaces: 1 space = child, 2 spaces = grandchild. The indent step is auto-detected.

■ **PASS** ■ **FAIL** **Notes:** _____

TEST 25 Advanced Controls

ProgressSteps — Step Indicator with Click

1 Build in Designer

1. Add Progress Steps — ID = pst_wizard
2. Set Steps = Account|Profile|Review|Confirm | Current Step = 1
3. Wire event — ■ Add → Event: Step_Click → Action: GoSub → Value: OnStepClick
4. Add Button — ID = btn_next | Caption = Next Step — wire Left_Click → OnNextStep
5. Add Label — ID = lbl_step | Caption = Step 1

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
pst_wizard	Step_Click	GoSub	OnStepClick
btn_next	Left_Click	GoSub	OnNextStep

How: In the Events tab, click ■ Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[pst_wizard.index]	Step number that was clicked (1-based) — "1", "2", "3", "4"

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
. Initialize the step counter when the page loads
:Home_OnEnter
SetVar "[StepNum]" "1"
Return

:OnStepClick
. [pst_wizard.index] = the step that was clicked (1-based)
SetVar "[StepNum]" "[pst_wizard.index]"
NWID_Set "Rectangle1" "label:lbl_step" "caption" "Clicked step [StepNum]"
Return

:OnNextStep
. Increment the step counter and update the control
Math "[StepNum]+1" "0" "[StepNum]"
NWID_Set "Rectangle1" "progressSteps:pst_wizard" "currentStep" "[StepNum]"
NWID_Set "Rectangle1" "label:lbl_step" "caption" "Step [StepNum]"
Return
```

Expected results:

- On load — [StepNum] = 1, first step is active
- Click step 3 (Review) — step highlights, label shows "Clicked step 3"
- Click Next Step — active step advances, completed steps show ✓
- Click Next Step past the last step — all four steps show as completed

Note: currentStep is 1-based (1 = first step). The Step_Click index is also 1-based.

Note: This approach uses a VNW variable to track state — no NWID_Get needed. Initialize [StepNum] in your page enter sub.

■ **PASS** ■ **FAIL** Notes: _____

TEST 26 Advanced Controls

CountdownTimer — Tick and Ended Events

1 Build in Designer

1. Add Countdown Timer — ID = cdt_timer
2. Set Seconds = 10 | Auto Start = off | Format = MM:SS
3. Wire events — ■ Add for Tick and Ended.
4. Add Button 1 — ID = btn_start | Caption = Start — wire Left_Click → OnStart
5. Add Button 2 — ID = btn_reset | Caption = Reset — wire Left_Click → OnReset
6. Add Label — ID = lbl_time | Caption = Ready

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
cdt_timer	Tick	GoSub	OnTick
cdt_timer	Ended	GoSub	OnTimerEnded
btn_start	Left_Click	GoSub	OnStart
btn_reset	Left_Click	GoSub	OnReset

How: In the Events tab, click ■ Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[cdt_timer.seconds]	Remaining time in seconds — updates every second

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
:OnStart
NWID_Set "Rectangle1" "countdownTimer:cdt_timer" "command" "start"
NWID_Set "Rectangle1" "label:lbl_time" "caption" "Running..."
Return

:OnReset
NWID_Set "Rectangle1" "countdownTimer:cdt_timer" "command" "reset"
NWID_Set "Rectangle1" "label:lbl_time" "caption" "Ready"
Return

:OnTick
NWID_Set "Rectangle1" "label:lbl_time" "caption" "Remaining: [cdt_timer.seconds]s"
Return
```

```
:OnTimerEnded  
NWID_Set "Rectangle1" "label:lbl_time" "caption" "Time is up!"  
Return
```

Expected results:

- Click Start — timer counts down from 10, label updates each second
- Timer reaches 0 — label shows "Time is up!"
- Click Reset — timer resets to 10, label shows "Ready"

Note: Use "command" with "start", "stop", or "reset" to control the timer. Use "seconds" to change the duration.

■ **PASS** ■ **FAIL** Notes: _____

TEST 27 Advanced Controls

ScrollingTicker — Marquee with Click Events

1 Build in Designer

1. Add Scrolling Ticker — ID = tkr_news
2. Set Items = Breaking News|Markets Rally|Weather Alert|Sports Update
3. Set Separator = • | Speed = 40 | Pause on Hover = on
4. Wire event — ■ Add → Event: Item_Click → Action: GoSub → Value: OnTickerClick
5. Add Button — ID = btn_update | Caption = Update Items — wire Left_Click → OnUpdateTicker
6. Add Label — ID = lbl_ticker | Caption = (empty)

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
tkr_news	Item_Click	GoSub	OnTickerClick
btn_update	Left_Click	GoSub	OnUpdateTicker

How: In the Events tab, click ■ Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[tkr_news.value]	Text of the clicked ticker item

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
:OnTickerClick
NWID_Set "Rectangle1" "label:lbl_ticker" "caption" "Clicked: [tkr_news.value]"
Return

:OnUpdateTicker
NWID_Set "Rectangle1" "scrollingTicker:tkr_news" "items" "Alert One|Alert Two|Alert Three"
Return
```

Expected results:

- Ticker scrolls continuously with items separated by •
- Hover over ticker — scrolling pauses
- Click an item — label shows "Clicked: Markets Rally"
- Click Update Items — ticker content changes

■ **PASS** ■ **FAIL** Notes: _____

TEST 28 Advanced Controls

Accordion — Collapsible Sections

1 Build in Designer

1. Add Accordion — ID = acd_faq | Height ~200px
2. Set Items (alternating title|body pairs): What is NWID?|A drag-and-drop UI designer.|How do I export?|Click the Export button.|Can I add pages?|Yes, click the + button in the Page Strip.
3. Set Open Index = 0 (first section open by default)
4. Wire events — ■ Add for Section_Open and Section_Close.
5. Add Label — ID = lbl_section | Caption = (empty)

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
acd_faq	Section_Open	GoSub	OnSectionOpen
acd_faq	Section_Close	GoSub	OnSectionClose

How: In the Events tab, click ■ Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[acd_faq.index]	Zero-based index of the section that was opened or closed

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
:OnSectionOpen
NWID_Set "Rectangle1" "label:lbl_section" "caption" "Opened section [acd_faq.index]"
Return

:OnSectionClose
NWID_Set "Rectangle1" "label:lbl_section" "caption" "Closed section [acd_faq.index]"
Return
```

Expected results:

- First section open on load ("What is NWID?")
- Click another header — it opens, previous closes
- Label updates with "Opened section 1" / "Closed section 0"

Note: Items are alternating title|body pairs. Multi Open = on allows multiple sections open simultaneously.

■ **PASS** ■ **FAIL** Notes: _____

TEST 29 Advanced Controls

SimpleChart — Bar/Line Chart with Click Events

1 Build in Designer

1. Add Simple Chart — ID = cht_sales | Height ~200px
2. Set Type = Bar Chart | Labels = Q1|Q2|Q3|Q4 | Values = 120|95|140|110
3. Set Show Grid = on | Show Values = on | Caption = Quarterly Sales
4. Wire event — ■ Add → Event: Bar_Click → Action: GoSub → Value: OnBarClick
5. Add Button — ID = btn_line | Caption = Switch to Line — wire Left_Click → OnSwitchLine
6. Add Button — ID = btn_data | Caption = New Data — wire Left_Click → OnNewData
7. Add Label — ID = lbl_chart | Caption = Click a bar

Events Tab — wire these subroutine names

Control ID	Event	Action	Subroutine
cht_sales	Bar_Click	GoSub	OnBarClick
btn_line	Left_Click	GoSub	OnSwitchLine
btn_data	Left_Click	GoSub	OnNewData

How: In the Events tab, click ■ Add for each row. Set Event to the event name, Action to GoSub, and enter the subroutine name in the Value field.

2 Export

Click **Export** in NWID. This generates two files: **runtime.html** and **runtime_subs.txt**. Copy both to your VNW publication folder.

Note: After importing the subs file in VNW, locate the stub subroutine(s) listed in the Events table above — they will appear at the top of the subs file with a Return and no code. That is where you write your plugin commands.

4 Script + Test

NWID sets these per-control variables before calling your subroutine:

Variable	Contains
[cht_sales.value]	Label Value of the clicked bar or point — e.g. "Q2 95"

Write these lines inside the stub subroutines in your VNW Subroutine Action:

Method 1 — Plugin commands:

```
:OnBarClick
NWID_Set "Rectangle1" "label:lbl_chart" "caption" "Clicked: [cht_sales.value]"
Return

:OnSwitchLine
NWID_Set "Rectangle1" "simpleChart:cht_sales" "chartType" "line"
Return

:OnNewData
NWID_Set "Rectangle1" "simpleChart:cht_sales" "values" "80|150|60|200"
NWID_Set "Rectangle1" "simpleChart:cht_sales" "labels" "Jan|Feb|Mar|Apr"
Return
```

Expected results:

- Bar chart shows Q1–Q4 with value labels above bars
- Click a bar — label shows "Clicked: Q2|95"
- Click Switch to Line — chart redraws as a line chart
- Click New Data — chart updates with new values and labels

Note: The value variable contains "label|value" separated by a pipe. Use SearchStr to parse them apart in VNW.

■ **PASS** ■ **FAIL** Notes: _____

Testing Checklist — Part 2

Check each item off as you complete it.

- TEST 21 ContextMenu — per-item GoSub, text controls excluded, icon tokens Notes: _____
- TEST 22 MediaPlayer — filename load, Play/Pause/Ended/Time_Update, command Notes: _____
- TEST 23 SearchBox — Search_Changed (debounced), Cleared Notes: _____
- TEST 24 TreeView — Node_Click, auto-detect indent, load by code Notes: _____
- TEST 25 ProgressSteps — Step_Click (1-based), VNW variable tracking, Next Step Notes: _____
- TEST 26 CountdownTimer — Tick, Ended, command start/stop/reset Notes: _____
- TEST 27 ScrollingTicker — Item_Click, update items, pause on hover Notes: _____
- TEST 28 Accordion — Section_Open/Close, alternating title|body Notes: _____
- TEST 29 SimpleChart — Bar_Click, chartType switch, values/labels update Notes: _____