

ContentBlocks v3.2 - VNW Delphi 11 Compatibility Fix Report

As of 2026-10-20 · Anthony C

Summary

ContentBlocks v3.1 crashed the new VisualNEO Win (rebuilt in Delphi 11) the moment `cbCreateContainer` ran. Two independent breaks in the classic plugin interface were found and fixed in July 2026, and v3.2 (`ContentBlocks_v3.2_HostCompat.pb`) runs on both the new host and the older Delphi 7 builds.

- 1. The host's `nbInterface` sub-function 8 (the design-time object picker) now crashes when called. All 14 call sites were rerouted through a new `CB_PickRectangle()` helper that uses PureBasic's `InputRequester` instead.
- 2. Memory ownership across pointer-swap calls flipped. The new host frees the plugin's input buffer itself and returns buffers from Delphi's own memory manager, so the old `GlobalFree-both-sides` contract became a double-free. Two compile switches now skip those frees.

The `Params[]` array contract, `nbInterface(7)` (object HWND lookup) and ANSI parameter strings survived the rebuild unchanged.

Background

ContentBlocks is a PureBasic-compiled `.nbp` DLL that gives VNW a rich scrollable content block (styled text and images inside a `Rectangle` host). Through v3.1 it was written against the classic VNW plugin SDK, whose conventions date from the Delphi 7 build of the host: ANSI `PChar` parameters, `GlobalAlloc/GlobalFree` ownership on both sides of the plugin boundary, and host services reached through `nbInterface(subFunction, pData)`.

When VNW shipped a release compiled in Delphi 11, `cbCreateContainer` brought the whole host down — at design time when the action editor opened, and at runtime when the action executed. The plugin kept working on the previous VNW, which pointed at the host, not the plugin's own logic. The source (`ContentBlocks.pb`) and a sample pub were handed over for analysis.

How it was diagnosed

The crash left no useful exception inside VNW, so the plugin was instrumented rather than reasoned about. Three diagnostic builds were compiled and run on the new host, each narrowing the fault from the trace the previous one produced:

Build	Change	What the log showed
HOSTFIX1	Added <code>#NB_DEBUG_LOG</code> tracing to <code>%TEMP%\ContentBlocks_debug.log</code> ,	Design-time crash occurred inside the

	hex dumps of host buffers, and host-ABI switches (<code>#NB_HOST_UNICODE</code> , <code>#NB_FREE_HOST_RETURNED</code> , <code>#NB_FREE_OLD_PARAMS</code>)	<code>nbInterface(8)</code> call itself; parameter strings were still ANSI, not UTF-16
HOSTFIX2	Object picker bypassed via <code>CB_PickRectangle()</code> ; design time now survived	Runtime crash moved to the first pointer-swap call: the plugin's own input buffer was freed, then the host's swapped-in pointer was freed through <code>GlobalFree_</code>
HOSTFIX3	Added <code>#NB_FREE_SWAPPED_INPUT = 0</code> and <code>#NB_FREE_HOST_RETURNED = 0</code> with guarded helper procedures	Clean run at design time and runtime; no further faults

The release build then set `#NB_DEBUG_LOG = 0` so the logging compiles out entirely.

Root cause 1 - the object picker sub-function

`nbInterface(8)` is the host's rectangle-only object selector: at design time it shows the user a list of Rectangle objects on the current page and writes the chosen name back. ContentBlocks called it from 14 action editors (every `cbCreate*` and `cbSet*` dialog that needs a host rectangle). On the Delphi 11 host the call never returns — VNW crashes inside it, so the plugin's action editor could not even open.

Fix. All 14 call sites now go through one helper, `CB_PickRectangle()`, controlled by a compile switch:

- `#NB_USE_OBJECT_PICKER = 0` (default, new host): shows a PureBasic `InputRequester` dialog where the user types the Rectangle name. Design-time setup no longer depends on that host service at all.
- `#NB_USE_OBJECT_PICKER = 1` (classic host): restores the native `nbInterface(8)` list.

If the picker's replacement sub-function is ever documented for the new host, a future v3.3 can bring the native list back behind the same switch.

Root cause 2 - pointer-swap memory ownership

Several host services return a result by *swapping the pointer*: the plugin passes `pData` (a `GlobalAlloc`'d buffer it owns), and on return `pData` points at a different, host-allocated buffer holding the answer. The classic Delphi 7 contract was symmetrical: the plugin freed its original input buffer (the host had not touched it) and, after reading the result, freed the host's returned buffer with `GlobalFree_`. ContentBlocks did exactly that in `CB_GetObjectHWND` (`nbInterface(7)`), the picker writeback, the old `Params[]` slot in `WriteParamString`, and the `nbGetVar` result.

On the Delphi 11 host both halves of that contract changed:

Buffer	Delphi 7 host	Delphi 11 host	Effect of the old code
Plugin's input	Left to the plugin;	Host frees it	Double-free →

buffer after a swap	plugin frees it	(takes ownership) when it swaps	heap corruption → runtime crash
Host's returned buffer	GlobalAlloc'd; plugin frees with GlobalFree_	Allocated by Delphi's memory manager (FastMM)	GlobalFree_ on a non-Global handle corrupts the heap

Fix. Two switches, both 0 for the new host, with guarded helpers so the frees compile out:

- `#NB_FREE_SWAPPED_INPUT = 0` — `CB_FreeSwappedInput(*ours, *now, tag)` skips freeing the plugin's own input buffer whenever the host swapped it.
- `#NB_FREE_HOST_RETURNED = 0` — never `GlobalFree_` a pointer the host allocated (`nbInterface 7/8` writeback, `nbGetVar` result, the old `Params[]` slot).

The cost is a few leaked bytes per call — bounded and negligible — against a guaranteed crash.

What survived the rebuild unchanged

- The `Params[]` contract: `nbEditAction/nbExecAction` still receive an array of 4-byte ANSI `PChar` slots, read with `ReadParam` and written with `WriteParamString`.
- `nbInterface(7)` (object `HWND` lookup by name) still works and is still used at runtime.
- ANSI parameter strings: the hex dumps showed no UTF-16 at the boundary, so `#NB_HOST_UNICODE` stays 0.
- The registration exports (`_nbInitPlugIn`, `_nbRegisterPlugIn`, `_nbRegisterInterfaceAccess`) and `nbAddAction` behave as before.

Compatibility switches

One source builds for either host. The release values target the Delphi 11 VNW; flip the three marked rows to build a classic Delphi 7 version.

Switch	Release (Delphi 11)	Classic (Delphi 7)	Purpose
<code>#NB_USE_OBJECT_PICKER</code>	0	1	0 = <code>InputRequester</code> dialog; 1 = native <code>nbInterface(8)</code> picker
<code>#NB_FREE_SWAPPED_INPUT</code>	0	1	0 = never free the plugin's own input buffer after a pointer swap (host owns it)
<code>#NB_FREE_HOST_RETURNED</code>	0	1	0 = never <code>GlobalFree_</code> host-allocated buffers
<code>#NB_HOST_UNICODE</code>	0	0	1 would switch the boundary to UTF-16; not needed on either host
<code>#NB_DEBUG_LOG</code>	0	0	1 appends a trace to <code>%TEMP%\ContentBlocks_debug.log</code> for troubleshooting

Release and lessons for other plugins

v3.2 “Host Compatibility Edition” was released on the VisualNEO forum in July 2026 after a smoke test of `cbCreateContainer` at design time and runtime on the new host. The version header in the `.pb` file documents the two interface changes, the surviving contracts and the three-switch recipe above, and the forum post led with the symptom rather than the plugin name so other developers searching for the crash would find it.

Any classic PureBasic plugin that does either of the following will hit the same two faults on the Delphi 11 host:

1. Calls `nbInterface(8)` for a design-time object list → replace with the plugin’s own dialog.
2. Frees buffers after a pointer-swap `nbInterface` call or an `nbGetVar` result → stop freeing anything the host has touched.

The same pattern was later applied to `SendMessageClickPlugin`, and the v3.2 teardown discipline (memory-frees-only in `DllMain` detach, `g_acceptingActions` gate, `WM_QUIT`-safe message drain) became the template for `NeoListBox`, `NeoSkin` and `NeoFreeze`.