

## Re: Try Finally...

**Source:** <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2004-10/1101.html>

---

**From:** VBDis (vbdis\_at\_aol.com)

**Date:** 10/28/04

Date: 28 Oct 2004 07:59:05 GMT

Im Artikel <2u8158F2706ubU1@uni-berlin.de>, Rob Kennedy <me3@privacy.net> schreibt:

>If SysUtils is not used, then system exceptions are not transformed into  
>Delphi exceptions, and so Delphi's try-except blocks don't handle them.

Okay, so one big part of Delphi SEH requires conversions for system exceptions, interrupts, assertions and other Delphi exceptions into Delphi SEH structures.

>(Delphi's try-finally blocks still execute, though.)

In my tests (D4) no Finally code was executed, the call to @Assert actually never returned and ended up in Halt instead, without triggering any Finally code. I.e. SEH is not working at all when the various exception conditions do not trigger the Delphi exception handling.

Consequently a replacement SEH should specify what it supports, at least:

- Delphi version
- which kinds of error conditions (by triggering Delphi exception handling)
- exceptions during exception handling (in exception handling code)
- debugging
- possibly non-standard behaviour of Except and Finally

More topics may need a specification (thread support...).

With regards to these many and possibly still incomplete parameters I doubt that a possible reduction of program size by at most 24KB justifies the implementation and use of a replacement SEH, besides for programs in a very restricted environment. Even in special cases an exclusion of the whole SEH seems to be the simplest solution, requiring nothing but not to use SysUtils.

So my question to Laura is:

What shall be the benefit of your SEH, as opposed to simply excluding SysUtils?

DoDi

comp.lang.pascal.delphi.misc: Re: Try Finally...

P.S.: @Assert shows another calling convention encoding, in addition to the (stdcall? cdecl?) convention with \_GetMem.