Forcing an ERROR_KEY_DELETED error when trying to open HKEY_CURRENT_USER\Software

devblogs.microsoft.com/oldnewthing/20250103-00

January 3, 2025

Last time, we wondered whether the `ERROR_KEY_DELETED` error code could be the result of the user logging off while a program was running. I noted that this is primarily an issue for services, though you can force it to happen for regular user-session applications.

It involves taking advantage of the `RegOverridePredefKey` which lets you redirect the registry operations performed by the current program to another registry key. The intention of this function was to allow you to override a registry root key like `HKEY_CLASSES_ROOT` so you can capture the registry keys written by an installer. But we're going to use it to force an `ERROR_KEY_DELETED` error from code that tries to open `HKEY_CURRENT_USER\Software`.

Note that this is just a parlor trick! There is no practical use for this demonstration.

```
#include <windows.h>
#include <wil/resource.h>
#include <wil/result_macros.h>

int main(int, char**)
{
    wil::unique_hkey originalSoftware;
    FAIL_FAST_IF_WIN32_ERROR(RegOpenKeyExW(HKEY_CURRENT_USER,
        L"Software", 0, KEY_READ, &originalSoftware));

    wil::unique_hkey originalContoso;
    DWORD disposition;
    FAIL_FAST_IF_WIN32_ERROR(RegCreateKeyExW(originalSoftware.get(),
        L"Contoso", 0, nullptr, REG_OPTION_VOLATILE,
        KEY_READ | KEY_WRITE, nullptr,
        &originalContoso, &disposition));
    if (disposition != REG_CREATED_NEW_KEY) {
        printf("Unexpected HKCU\\Software\\Contoso key\n");
        return 0;
    }

    FAIL_FAST_IF_WIN32_ERROR(RegOverridePredefKey(
        HKEY_CURRENT_USER, originalContoso.get()));

    FAIL_FAST_IF_WIN32_ERROR(
        RegDeleteKeyW(originalSoftware.get(), L"Contoso"));

    wil::unique_hkey key;
    LSTATUS error = RegOpenKeyExW(HKEY_CURRENT_USER, L"Software", 0, KEY_READ, &key);
    if (error == ERROR_KEY_DELETED)
    {
        printf("It happened!\n");
    } else {
        printf("It didn't happen.\n");
    }

    return 0;
}
```

First, we create a key to the original `HKEY_CURRENT_USER\Software`, and then create a `Contoso` subkey under it. If there was already a `Contoso` subkey, then we bail out because our trick won't work. (We'll have to pick another key to use as our sacrificial lamb.)

After creating the `Contoso` subkey, we pass it to `RegOverridePredefKey` to make it the new `HKEY_CURRENT_USER`. Any future references to `HKEY_CURRENT_USER` will use our `originalContoso` key instead of the original `HKEY_CURRENT_USER`. (Note that our original keys still refer to their original unredirected versions.)

Our final step for preparing the trick is deleting the `Contoso` key from the original `HKEY_CURRENT_USER\Software`. Since we had redirected `HKEY_CURRENT_USER` to refer to the original `HKEY_CURRENT_USER\Software\Contoso` key, deleting that key means that `HKEY_CURRENT_USER` is now a reference to a deleted key.

The next line of code is our victim. It tries to open `HKEY_CURRENT_USER\Software`, which is an attempt to reference a `Software` subkey under the original now-deleted `HKEY_CURRENT_USER\Software\Contoso` key. Since the key has been deleted, the error is `ERROR_KEY_DELETED`.

I’m not saying this is what was causing the original code to get an `ERROR_KEY_DELETED` error when trying to open `HKEY_CURRENT_USER\Software`. I’m just saying that this is one way it can happen, though it is extremely contrived.