Diagnosing why you cannot create a stable subkey under a volatile parent key

devblogs.microsoft.com/oldnewthing/20170525-00

May 25, 2017

Raymond Chen

A customer encountered crashes in their program's initialization code. They weren't able to reproduce the problem in-house, but their failure logs suggested it was coming from here:

```
var settingsKey =
    Registry.CurrentUser.CreateSubKey(
        "Software\\Contoso\\Common Settings\\Drawing Preferences");
```

The call was failing with `System.IO.IOException: Cannot create a stable subkey under a volatile parent key`. The corresponding Win32 error code is `ERROR_CHILD_MUST_BE_VOLATILE`.

First of all, what does this error mean?

This error means exactly what is says: You cannot create a stable (non-volatile) subkey under a volatile parent key. All children of a volatile key must themselves be volatile.

Okay, but why is the parent key volatile?

We don't know for sure which key is the volatile parent, but it's one of `Software`, `Software\\Contoso`, or `Software\\Contoso\\Common Settings`. We can probably rule out `Software` since that key is pre-created by the system. That leaves the other two Contoso keys. But they are intended to hold persistent settings. Why would anybody create those keys as volatile? That would defeat the purpose of the keys.

Ah, but perhaps the parent keys were created volatile by mistake. An often-overlooked detail of the `RegCreateKeyEx` function (which is therefore also a detail of the `CreateSubKey` CLR method) is that if you ask for the key to be created as volatile, then the volatility applies to *all* keys created by the call. This means two things:

- If the key already exists, then its stability is unchanged. If it was volatile before, then it remains volatile. If it was stable before, then it remains stable.
- If the key doesn't already exist, then not only is the new key volatile, but the volatility also applies to any parent keys that didn't already exist.

By searching the code for any attempts to create volatile keys, we found this one that seemed suspicious:

```
var sessionSettings =
    Registry.CurrentUser.CreateSubKey(
        "Software\\Contoso\\Common Settings\\Current Session",
        RegistryOptions.Volatile);
```

The intent of this code was to create a volatile `Current Session` key to hold the user's temporary settings that should be discarded when the user logs off. However, if the `Contoso\\Common Settings` key doesn't yet exist, this will create not only a volatile `Current Session`, but also a volatile `Common Settings` key, and possibly even a volatile `Contoso` key!

My theory as to what is going on is that the failures are occurring on machines where the call to create the `Current Session` key (1) occurs when the `Common Settings` key does not already exist, (2) comes before the call to create the `Drawing Preferences` key, and (3) ends up being the call that creates the `Common Settings` key as a volatile key. One possibility is that this is the first time any program developed by Contoso has been run by this user, which means that none of the Contoso keys exist at the point the program starts. Another possibility is that the user, in a perhaps misguided attempt to fix a problem with a Contoso-developed program, deleted the entire `Common Settings` key, or possibly even the entire `Contoso` key.

The code to create the `Current Session` key should do so in two steps. First, create the stable parent key. Second, create the volatile subkey.

```
var commonSettings =
    Registry.CurrentUser.CreateSubKey(
        "Software\\Contoso\\Common Settings");
var sessionSettings = commonSettings.CreateSubKey(
        "Current Session",
        RegistryOptions.Volatile);
```

(Translating this to raw Win32 is left as an exercise for the reader.)

Raymond Chen

Follow