Why does SHGetKnownFolderPath fail when impersonating?

devblogs.microsoft.com/oldnewthing/20160601-00

June 1, 2016

Raymond Chen

A customer was having trouble with the `SHGetKnownFolderPath` function.

> We are calling the `SHGetKnownFolderPath` function. from a service while impersonating a user, but it returns `E_ACCESSDENIED`. It's failing both for `FOLDERID_ProgramData` and `FOLDERID_RoamingAppData`:
>
> ```
> hr = SHGetKnownFolderPath(
>         FOLDERID_ProgramData, // folder ID
>         0,                    // flags
>         nullptr,              // token
>         &path);           // result
> ```

Before we could reply, the customer followed up:

> We found that if we save the token that is being used for impersonation and pass it to the `SHGetKnownFolderPath` function. then it works. But we are not sure why is it working now since the call was already being made while impersonating, and the documentation for the `SHGetKnownFolderPath` function says that if you want to get the known folder path for the current user, then you pass `NULL` as the token.

Remember that the default answer to "Does this work while impersonating?" is No. When the `SHGetKnownFolderPath` function says "the current user", it roughly means "the user under whose identity the process is running", but more importantly, it means the user whose registry is mapped as `HKEY_CURRENT_USER`, which, as we saw earlier, is a very tricky proposition for a service that impersonates.

In this case, what's probably happening is that the service is running as something like SYSTEM, and `HKEY_CURRENT_USER` points to SYSTEM's registry, and the impersonated user does not have access to SYSTEM's registry, hence `E_ACCESSDENIED`.

The backstory is that the `SHGetKnownFolderPath` function (and its close relatives like `SHGetSpecialFolderPath`) are overwhelmingly used by normal applications that do no impersonation, so that's the use case they are optimized for. if you pass `NULL` as the token,

then the functions will read from `HKEY_CURRENT_USER` and cache the results. If the thread is impersonating, but you fail to pass the impersonation token to the `SHGetKnownFolder-Path` function, then it might return the (wrong) cached value, or it might try to read the value from the (wrong) `HKEY_CURRENT_USER` hive. Whatever happens, it's probably going to be wrong. To say, "No, don't use any of your fancy optimizations for normal applications, because I'm not a normal application. Get the known folder path for this specific user by reading from that user's registry."

You might say, "Well, the `SHGetKnownFolderPath` function should auto-detect whether it is being called when impersonating and compensate accordingly." Maybe, but that means introducing an expensive test to a hot code path to cover a rare case. that people shouldn't expect to work anyway because the default answer to "Does this work while impersonating?" is No. Instead, the extra work of dealing with impersonation is transferred to the people who want to get this information while impersonating.

Bonus chatter: Even if you pass the correct token to the `SHGetKnownFolderPath` function, you may still run afoul of other requirements. In particularly, note the sentence

> In addition to passing the user's *hToken*, the registry hive of that specific user must be mounted.

If the registry hive is not mounted, then there is no way to look up the information in the registry because, well, it's not in the registry. You can use the `LoadUserProfile` function to load the profile into the registry. Just remember to call `UnloadUserProfile` when you're done. (See the documentation for details.)

Raymond Chen

Follow