FULL FORCE AHEAD

NIDays 2019

NATIONAL INSTRUMENTS™

CARYA

Share Your Experiences at #NIDays

Leverage Windows Security

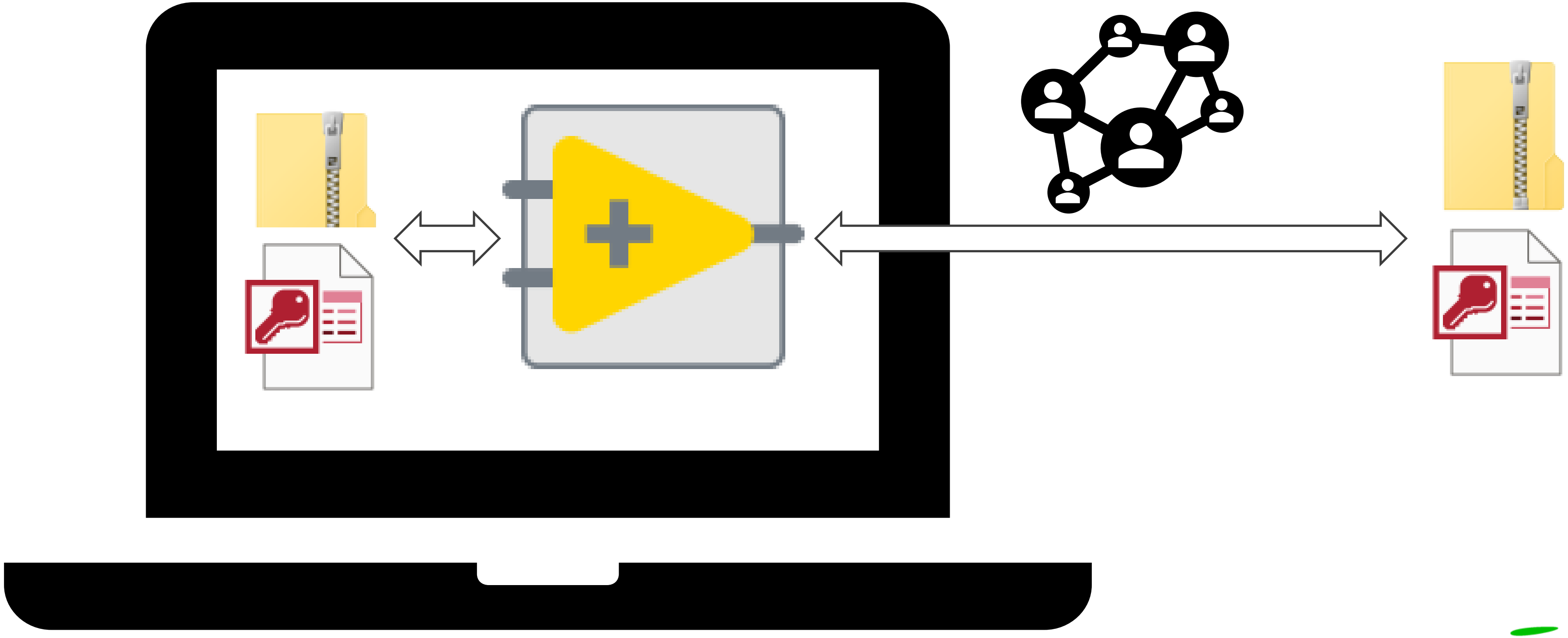
Wiebe Walstra - Carya - Netherlands

Agenda

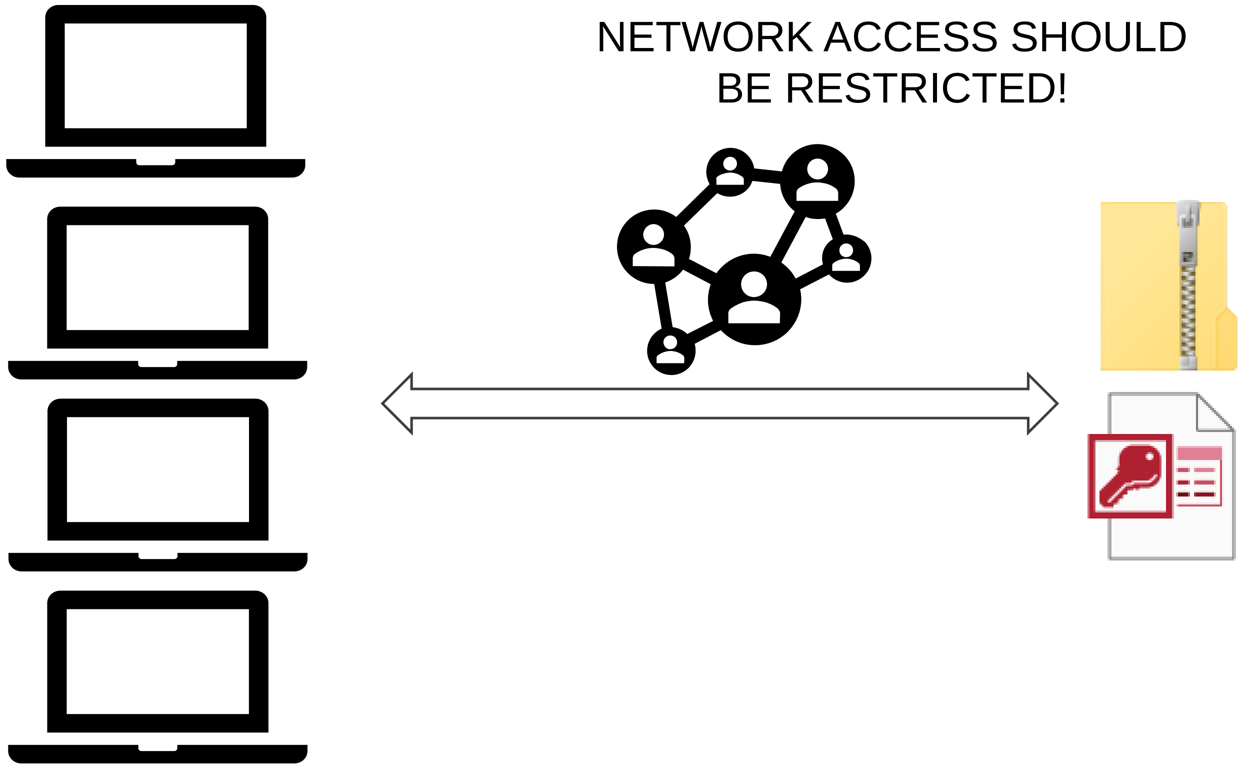
- Problem
- Early solutions
  - Not working
  - And why
- Solution (working)
  - Ins & outs

Problem

- Customer…
- Started ~2000
    - (yes that is 19 years ago)



NETWORK ACCESS SHOULD BE RESTRICTED!



Early solution (1)

- NET USE (CL)
- NET USE drive "Computer\Share\volume" /USER:[Domain]\user password /PRESISTENT:no
- NET USE Q “server\app\" /USER:Operator OperatorPW /PRESISTENT:no

- NET USE when needed
- mpr.dll \ WNetCancelConnectionA

Early solution (1) not working

- Map exists until deleted
  - Access during that time
  - Access after crash
- Windows only allows 1 mapping to a share…

Early solution (2)

- Start application as another user

- CPAU (CL)

Early solution (2) not working

- Entire application gets rights
- Stopped working with Windows 7

Early solution (3)

- Start application as another user
- RunAsSPC (CL) (Not free)

Early solution (3) not working

- Entire application gets rights

- Behaviour changed in Windows 10
  - Local user loses rights

Solution?

- Google \ MSDN \ DLLs

Solution

- Google \ MSDN \ **DLLs**

- SetThreadToken (MSDN)

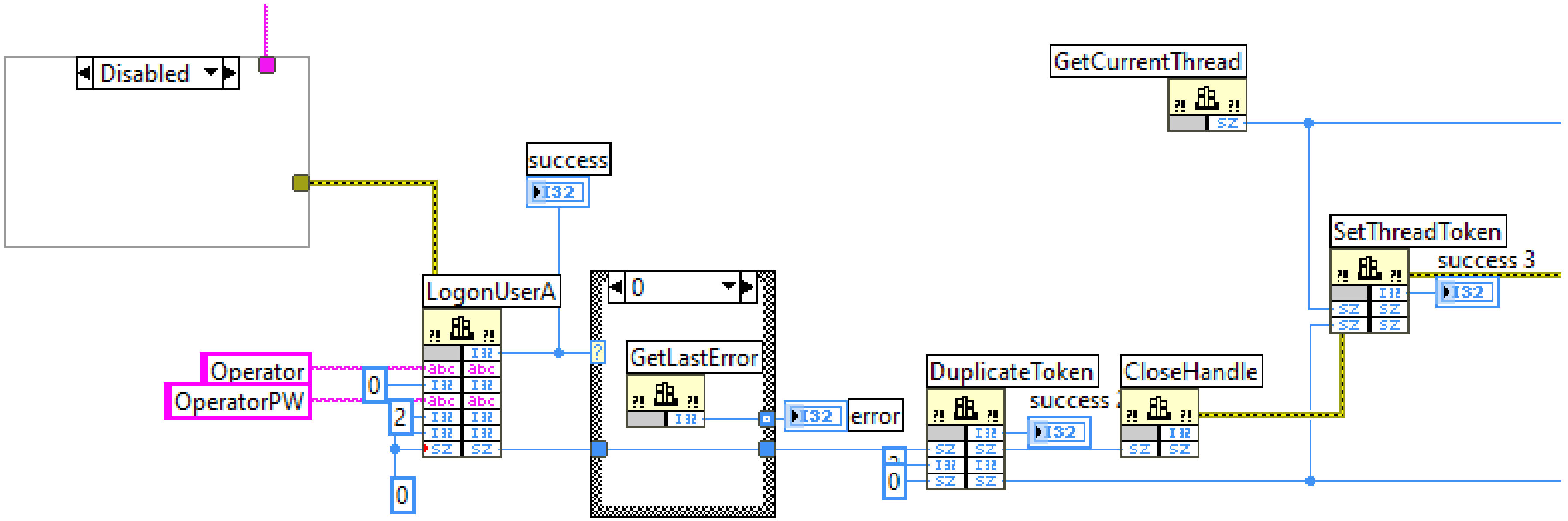
- The **SetThreadToken** function assigns an impersonation token to a thread.
- The function can also cause a thread to stop using an impersonation token.

- BOOL SetThreadToken(PHANDLE Thread, HANDLE Token);

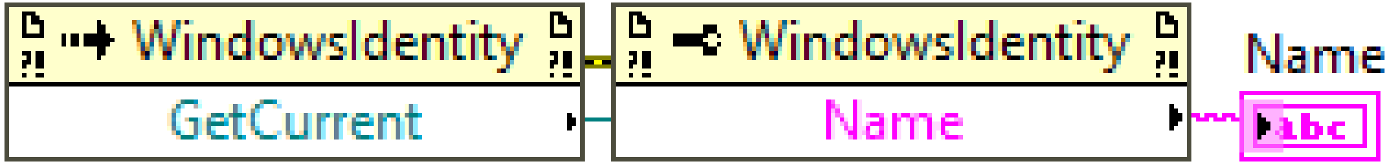
- LogonUser (MSDN)

- The **LogonUser** function attempts to log a user on to the local computer.
- The local computer is the computer from which **LogonUser** was called.
- You cannot use **LogonUser** to log on to a remote computer.

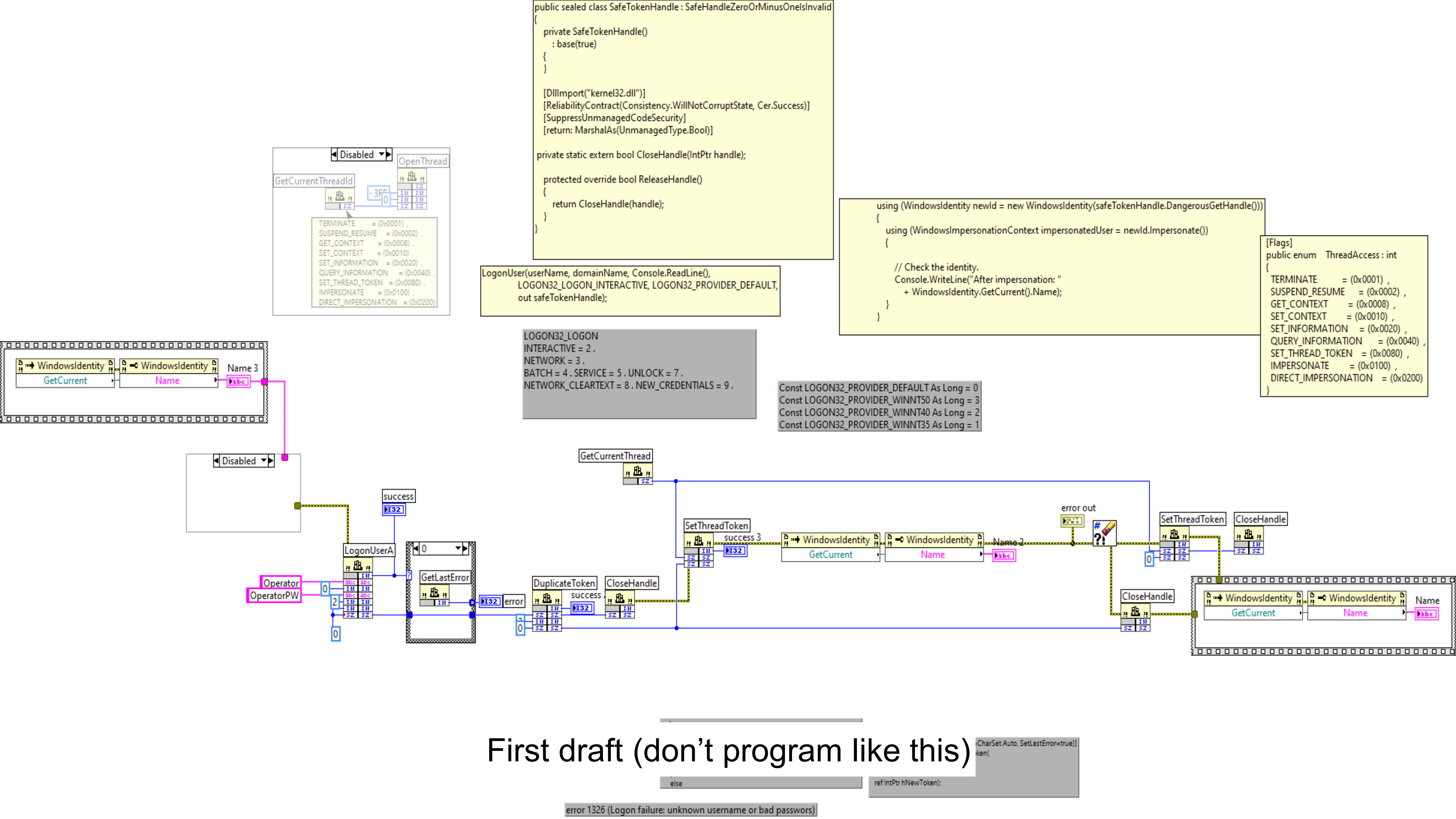
- SetThreadToken
- GetCurrentThread
- GetCurrentThreadId
- OpenThread
- LogonUserA
- (GetLastError)
- DuplicateToken
- CloseHandle



First draft (don’t program like this)


WindowsIdentity
GetCurrent
WindowsIdentity
Name
Name
abc

First draft (don’t program like this)



```
public sealed class SafeTokenHandle : SafeHandleZeroOrMinusOneIsInvalid
{
    private SafeTokenHandle()
      : base(true)
    {
    }

    [DllImport("kernel32.dll")]
    [ReliabilityContract(Consistency.WillNotCorruptState, Cer.Success)]
    [SuppressUnmanagedCodeSecurity]
    [return: MarshalAs(UnmanagedType.Bool)]
  private static extern bool CloseHandle(IntPtr handle);

    protected override bool ReleaseHandle()
    {
      return CloseHandle(handle);
    }
}
```

```
LogonUser(userName, domainName, Console.ReadLine(),
          LOGON32_LOGON_INTERACTIVE, LOGON32_PROVIDER_DEFAULT,
          out safeTokenHandle);
```

```
LOGON32_LOGON
INTERACTIVE = 2 .
NETWORK = 3 .
BATCH = 4 . SERVICE = 5 . UNLOCK = 7 .
NETWORK_CLEARTEXT = 8 . NEW_CREDENTIALS = 9 .
```

```
Const LOGON32_PROVIDER_DEFAULT As Long = 0
Const LOGON32_PROVIDER_WINNT50 As Long = 3
Const LOGON32_PROVIDER_WINNT40 As Long = 2
Const LOGON32_PROVIDER_WINNT35 As Long = 1
```

```
    using (WindowsIdentity newId = new WindowsIdentity(safeTokenHandle.DangerousGetHandle()))
    {
      using (WindowsImpersonationContext impersonatedUser = newId.Impersonate())
      {

        // Check the identity.
        Console.WriteLine("After impersonation: "
          + WindowsIdentity.GetCurrent().Name);
      }
    }
```

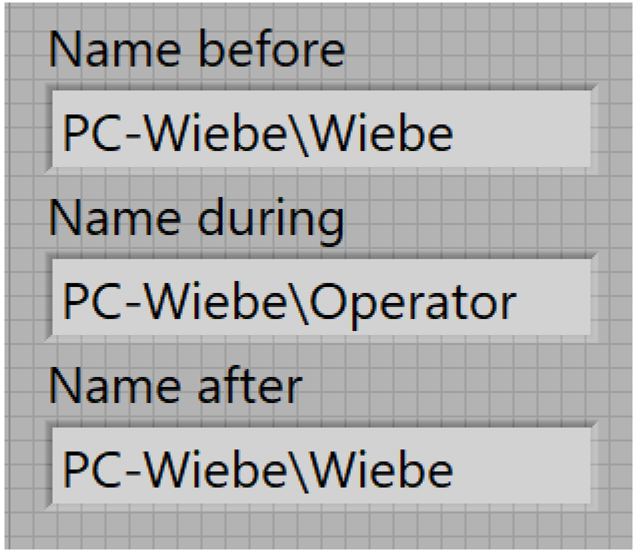
```
[Flags]
public enum    ThreadAccess : int
{
  TERMINATE          = (0x0001) ,
  SUSPEND_RESUME     = (0x0002) ,
  GET_CONTEXT        = (0x0008) ,
  SET_CONTEXT        = (0x0010) ,
  SET_INFORMATION    = (0x0020) ,
  QUERY_INFORMATION  = (0x0040) ,
  SET_THREAD_TOKEN   = (0x0080) ,
  IMPERSONATE        = (0x0100) ,
  DIRECT_IMPERSONATION  = (0x0200)
}
```

```
TERMINATE          = (0x0001) ,
SUSPEND_RESUME     = (0x0002) ,
GET_CONTEXT        = (0x0008) ,
SET_CONTEXT        = (0x0010) ,
SET_INFORMATION    = (0x0020) ,
QUERY_INFORMATION  = (0x0040) ,
SET_THREAD_TOKEN   = (0x0080) ,
IMPERSONATE        = (0x0100) ,
DIRECT_IMPERSONATION  = (0x0200)
```

```
=CharSet.Auto, SetLastError=true)]
ken(
ref IntPtr hNewToken);
```

```
else
```

error 1326 (Logon failure: unknown username or bad passwors)


Name before

PC-Wiebe\Wiebe

Name during

PC-Wiebe\Operator

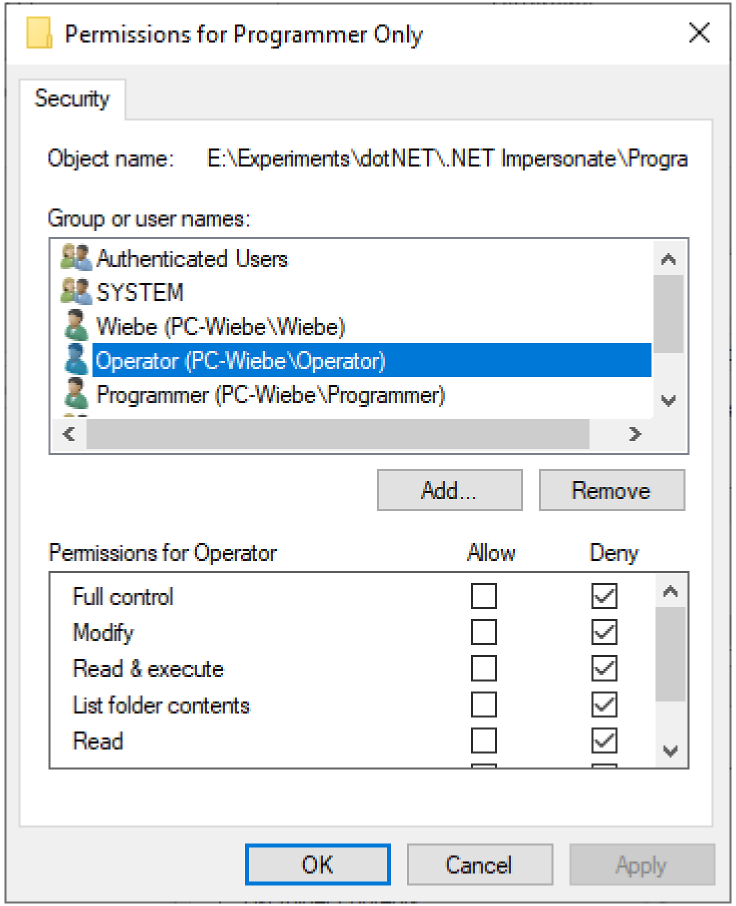
Name after

PC-Wiebe\Wiebe

My account (admin)

Operator

Programmer



Permissions for Programmer Only

Security

Object name: E:\Experiments\dotNET\.NET Impersonate\Progra

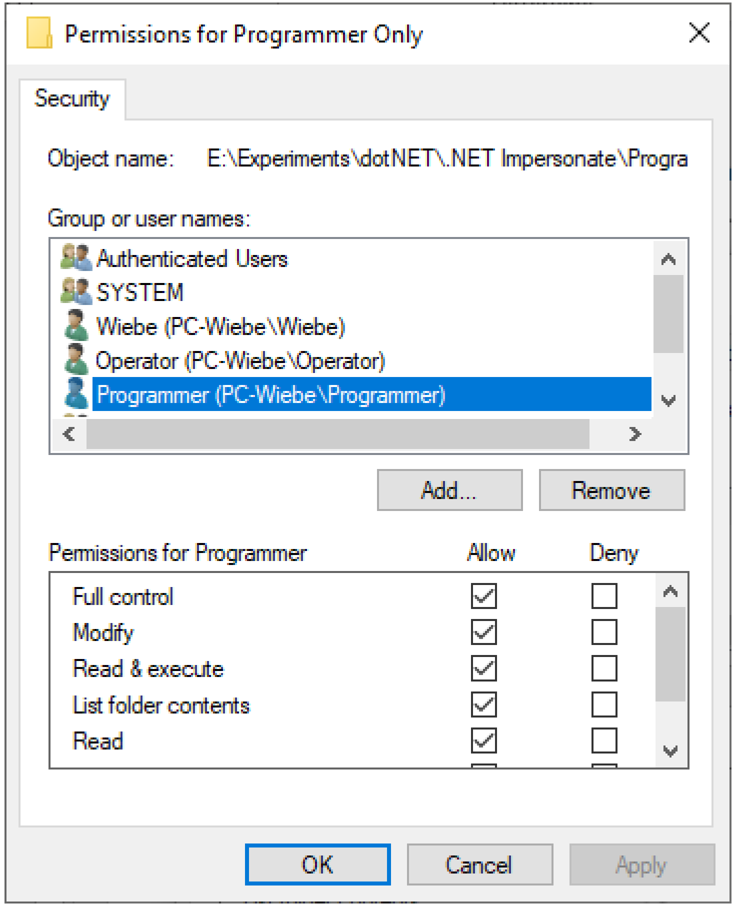
Group or user names:

- Authenticated Users
- SYSTEM
- Wiebe (PC-Wiebe\Wiebe)
- Operator (PC-Wiebe\Operator)
- Programmer (PC-Wiebe\Programmer)

Add... Remove

Permissions for Operator	Allow	Deny
Full control	☐	✓
Modify	☐	✓
Read & execute	☐	✓
List folder contents	☐	✓
Read	☐	✓

OK Cancel Apply



Permissions for Programmer Only

Security

Object name: E:\Experiments\dotNET\.NET Impersonate\Progra

Group or user names:

- Authenticated Users
- SYSTEM
- Wiebe (PC-Wiebe\Wiebe)
- Operator (PC-Wiebe\Operator)
- Programmer (PC-Wiebe\Programmer)

Add... Remove

Permissions for Programmer	Allow	Deny
Full control	✓	☐
Modify	✓	☐
Read & execute	✓	☐
List folder contents	✓	☐
Read	✓	☐

OK Cancel Apply

NATIONAL INSTRUMENTS

CARYA

INS & OUTS

(AKA the gory details)

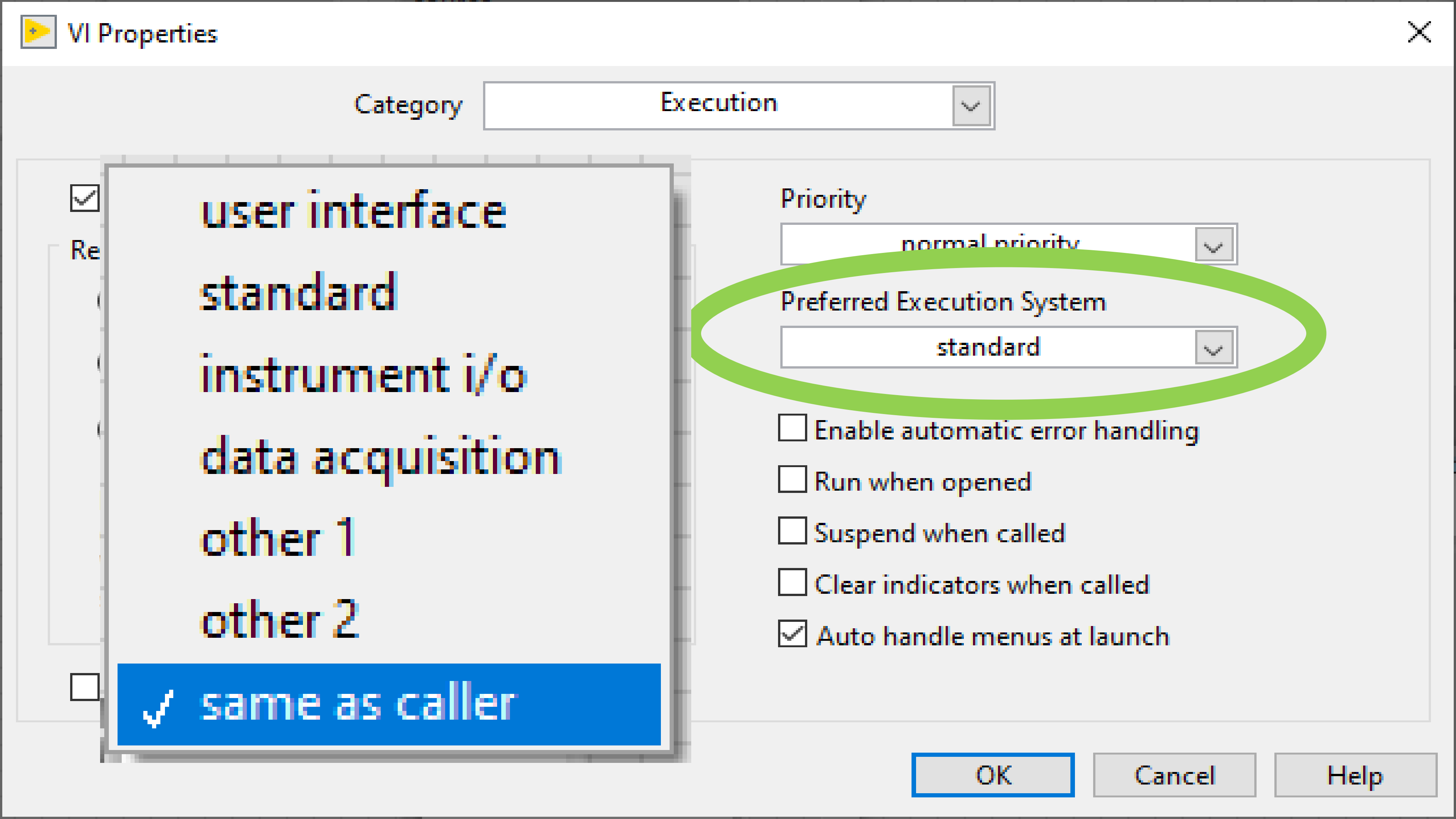
Ins & outs

- SetThreadToken

- The **SetThreadToken** function assigns an impersonation token to a **thread**.
- The function can also cause a **thread** to stop using an impersonation token.

- Makes sense in C\C++
- Not so much in LabVIEW

- But we can still use it!



VI Properties

Category: Execution

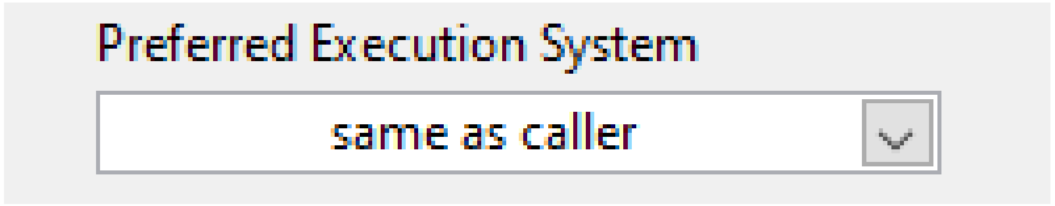
Re

user interface
standard
instrument i/o
data acquisition
other 1
other 2
✓ same as caller

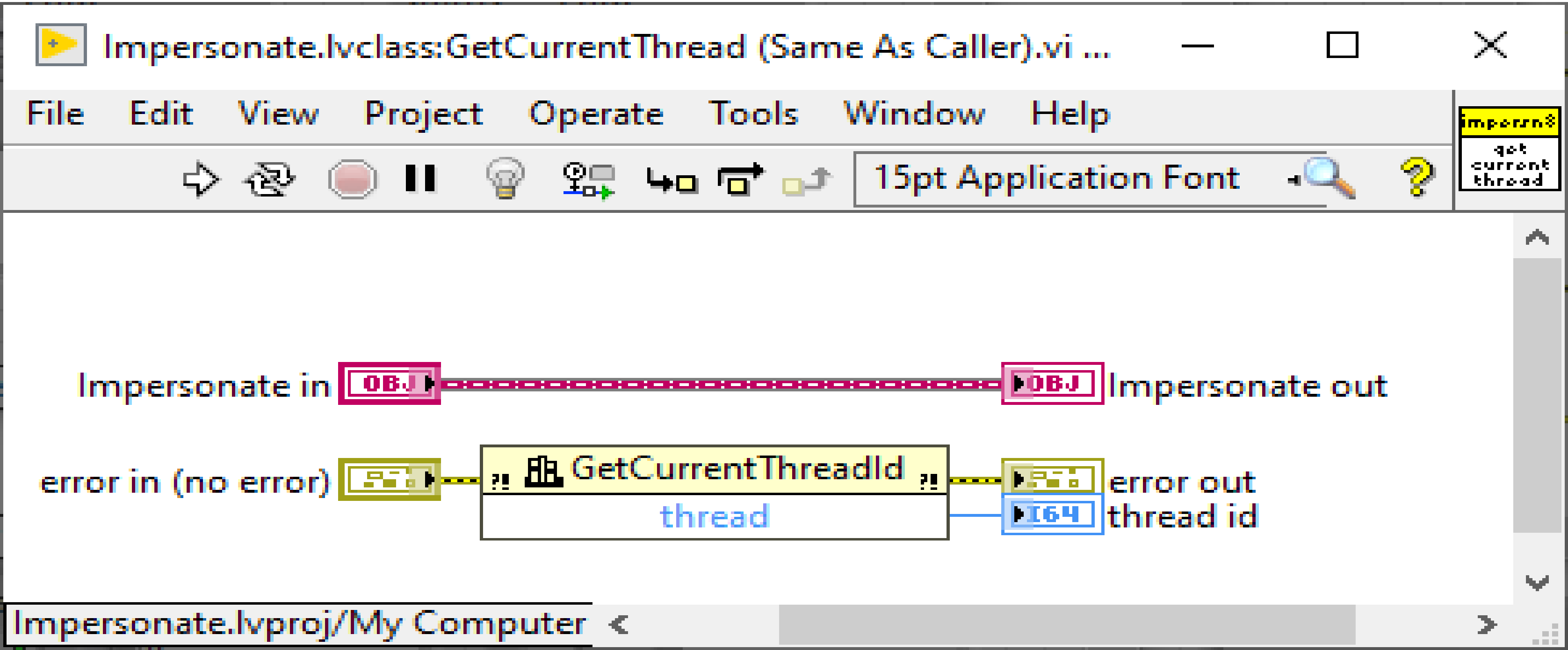
Priority: normal priority

Preferred Execution System: standard

- [ ] Enable automatic error handling
- [ ] Run when opened
- [ ] Suspend when called
- [ ] Clear indicators when called
- [x] Auto handle menus at launch


Preferred Execution System

same as caller

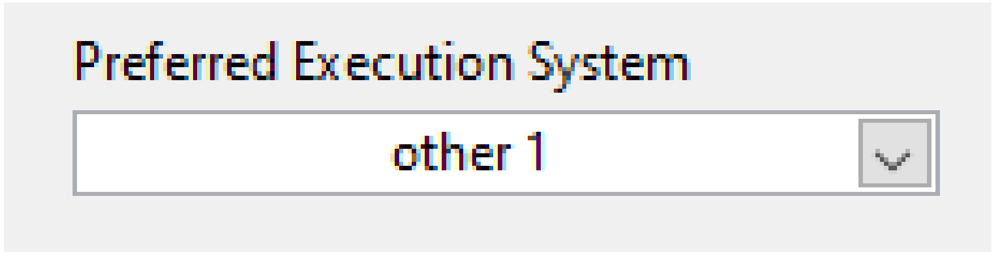

Impersonate.lvclass:GetCurrentThread (Same As Caller).vi ...

Impersonate in → Impersonate out

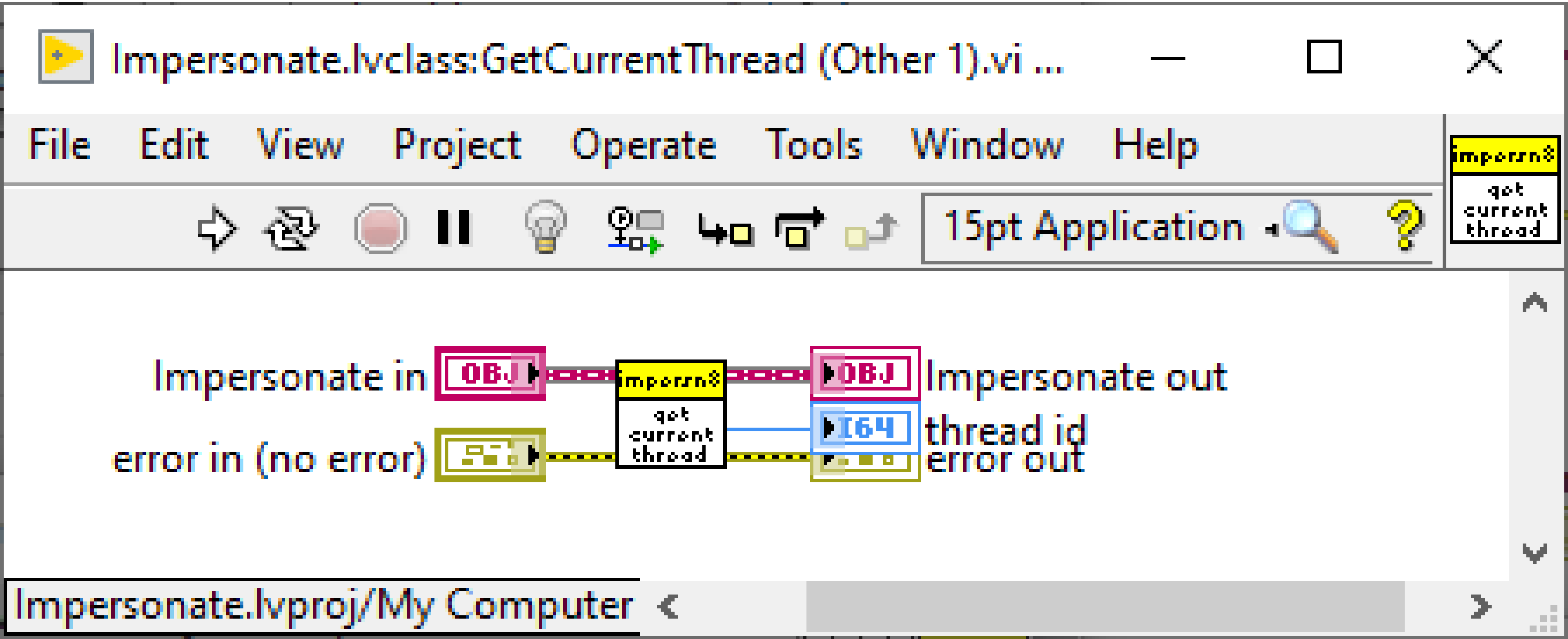
error in (no error) → GetCurrentThreadId → error out

thread → thread id

Impersonate.lvproj/My Computer


Preferred Execution System

other 1


Impersonate.lvclass:GetCurrentThread (Other 1).vi ...

Impersonate in

error in (no error)

Impersonate out

thread id

error out

Impersonate.lvproj/My Computer

GetCurrentThread (Data Acquisition).vi

GetCurrentThread (Instrument IO).vi

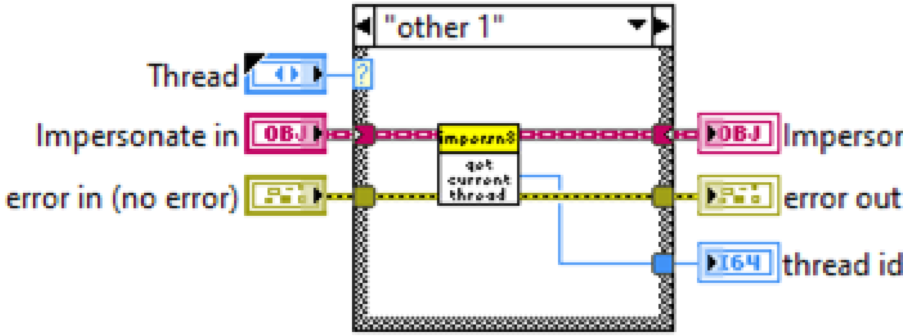
GetCurrentThread (Other 1).vi

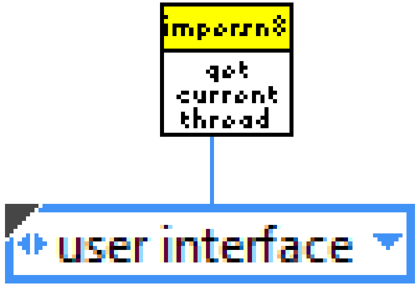
GetCurrentThread (Other 2).vi

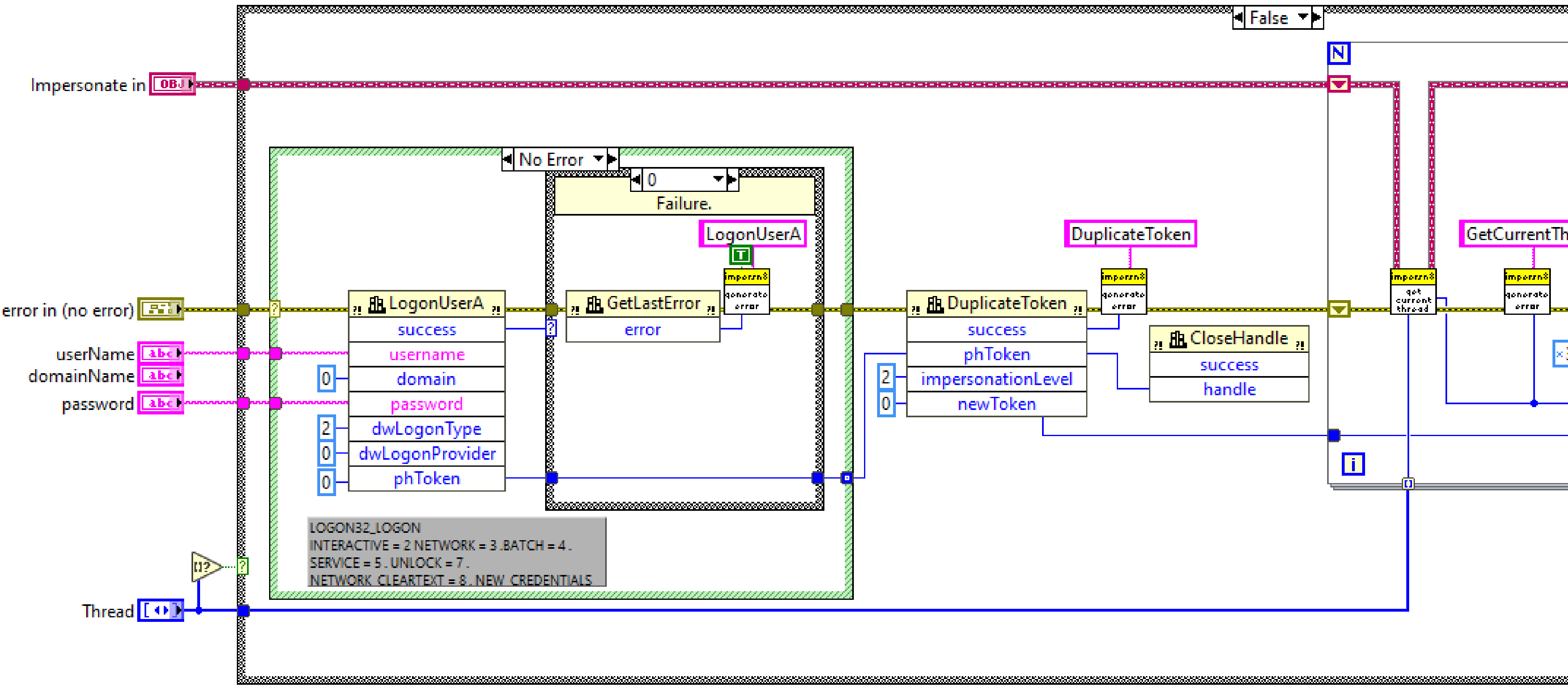
GetCurrentThread (Same As Caller).vi

GetCurrentThread (Standard).vi

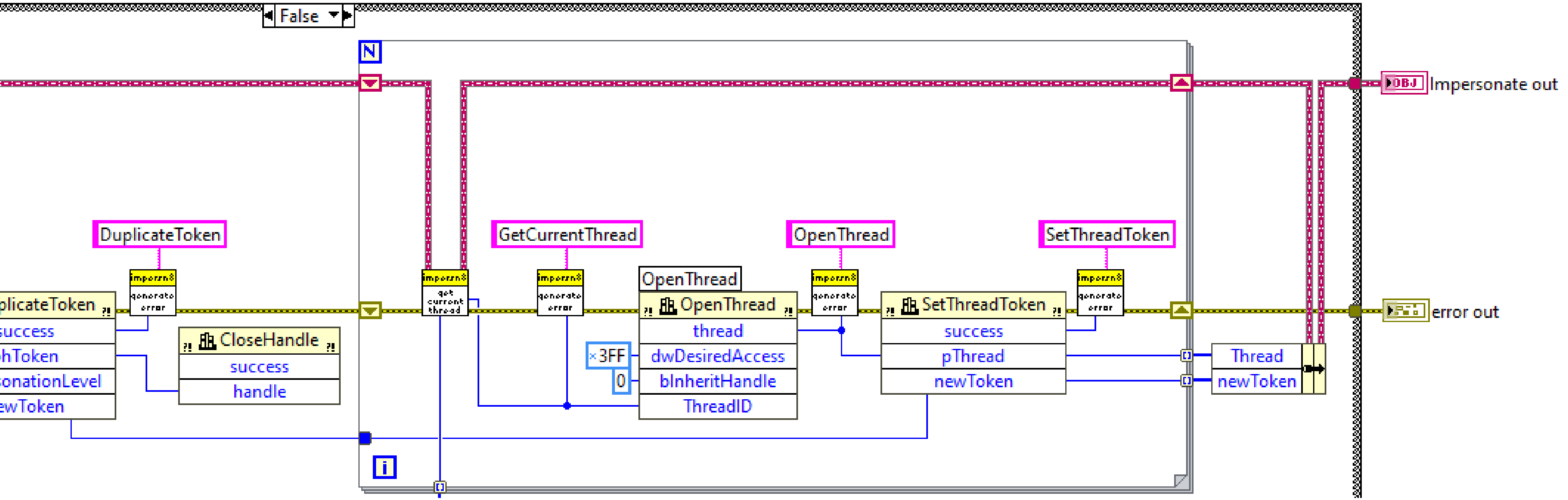
GetCurrentThread (User Interface).vi





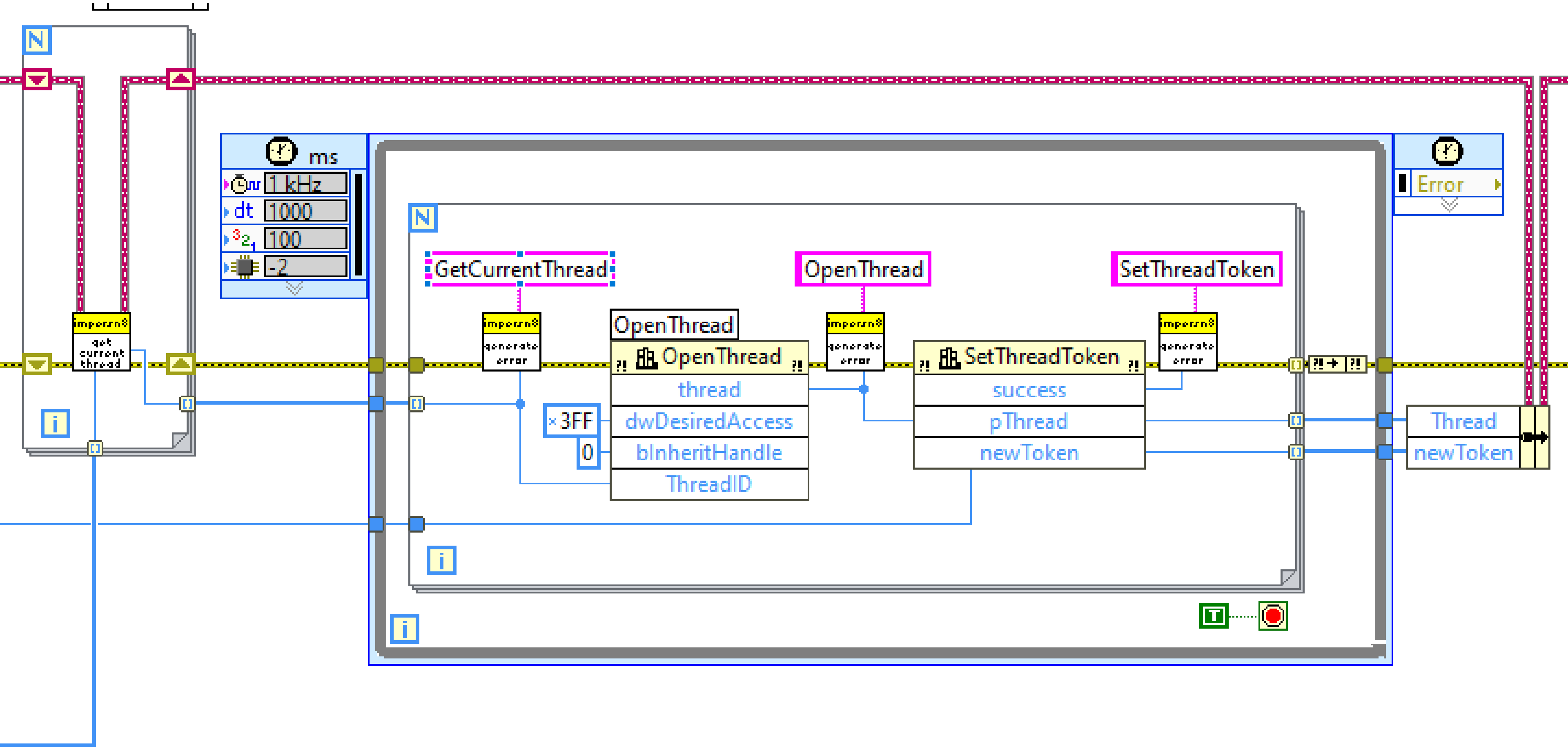


LOGON32_LOGON
INTERACTIVE = 2 NETWORK = 3 BATCH = 4
SERVICE = 5 UNLOCK = 7
NETWORK_CLEARTEXT = 8 NEW_CREDENTIALS

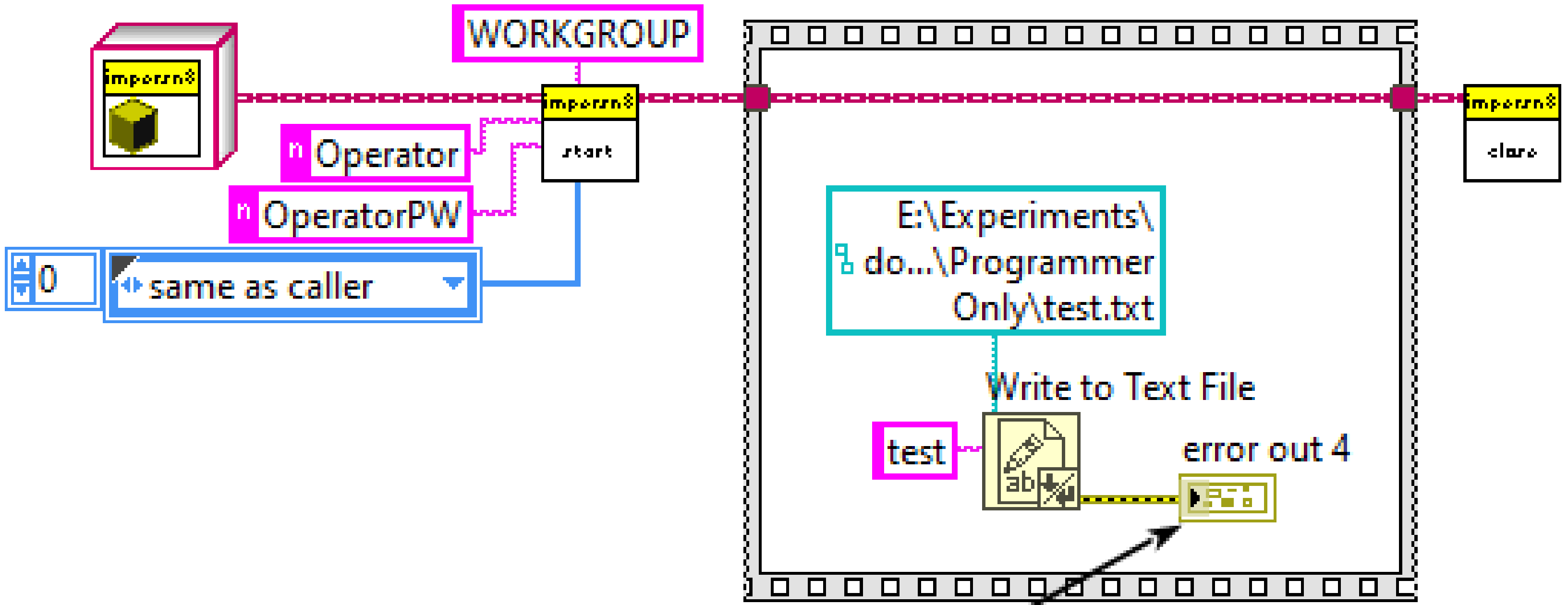


Order becomes important!

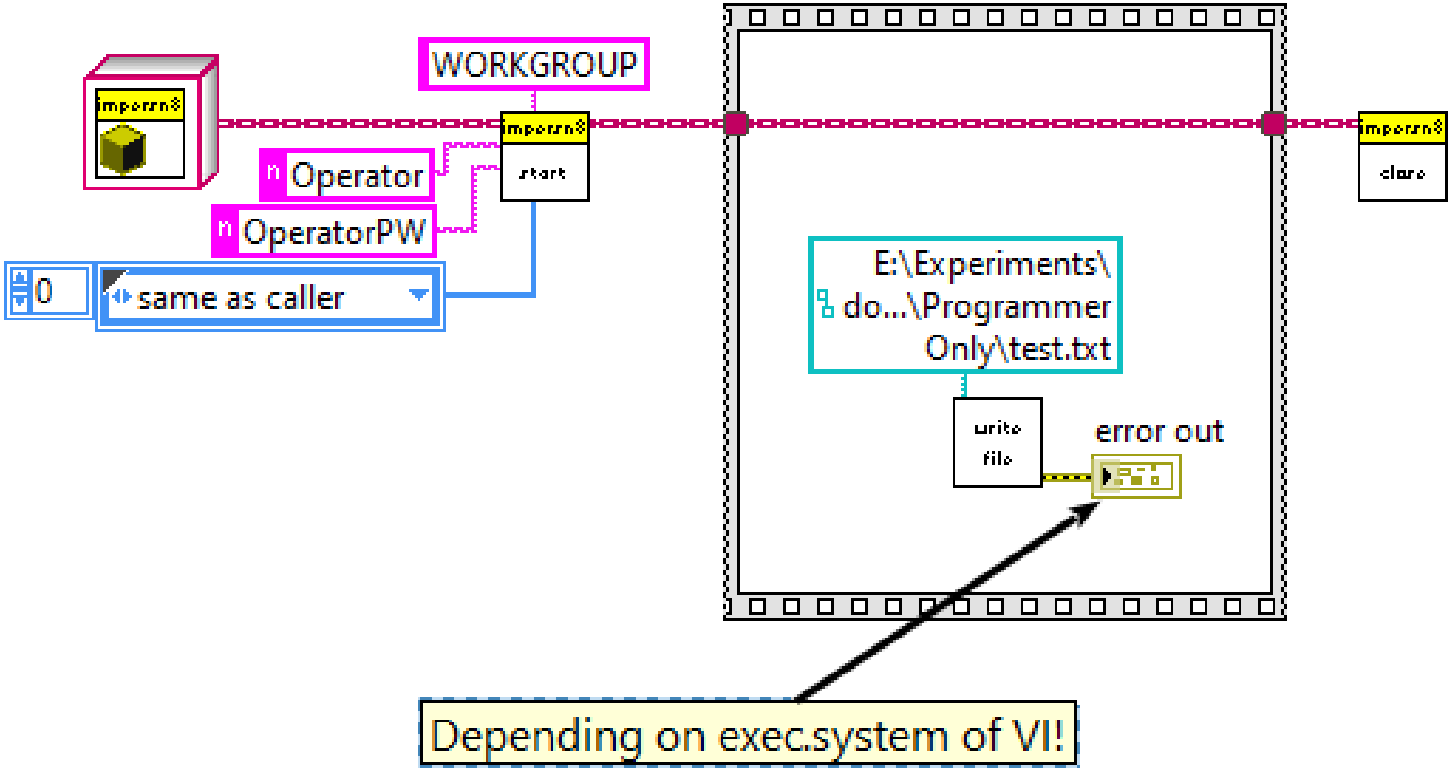
One more way to control thread



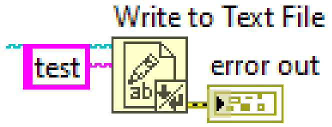
ms
1 kHz
dt 1000
100
-2
Error
GetCurrentThread
OpenThread
SetThreadToken
imporsn8
get current thread
generate error
OpenThread
thread
dwDesiredAccess
bInheritHandle
ThreadID
x3FF
0
SetThreadToken
success
pThread
newToken
Thread
newToken
N
i
T



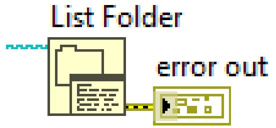
LabVIEW: (Hex 0x8) File permission error. You do not have the correct permissions for the file.



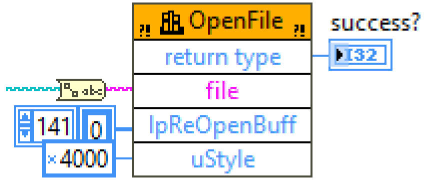
Depending on exec.system of VI!



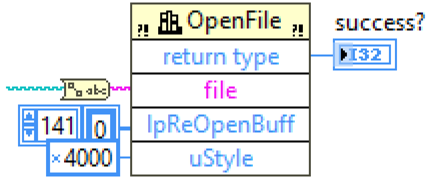
> VI thread (same as caller)



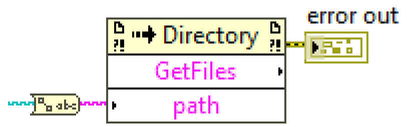
> User interface thread



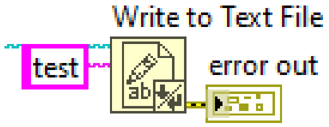
> User interface thread



> VI thread (same as caller)



> VI thread (same as caller)

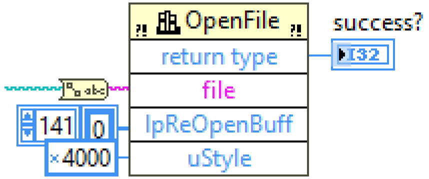


-> VI thread (same as caller)

then it's unclear?

not one of the standard threads

unless DLL (Other) is running in parallel



> VI thread (same as caller)

Concerns

- More sophisticated VIs?
- Credentials need to be on Windows?
- Networking is slightly different?
- Is it safe (plaintext pw)?
- Passwords might rotate?
- LV Versions? Windows versions? NXG?

Conclusion

- ‘It works’
- Not carefree
- No other options?

Stay Connected During and After NIDays

ni.com/community

facebook.com/NationalInstruments

twitter.com/niglobal

youtube.com/nationalinstruments