James Forshaw (@tiraniddo)

Infiltrate 2017

https://pixabay.com/en/hong-kong-city-urban-skyscrapers-1990268/

Agenda

- Component Object Model (COM) Internals
- Attacking COM
  - Enumerating attack surface for EoP
  - Reverse engineering COM components
- Bugs and “Features”

We’ll be using my OleViewDotNet tool throughout.

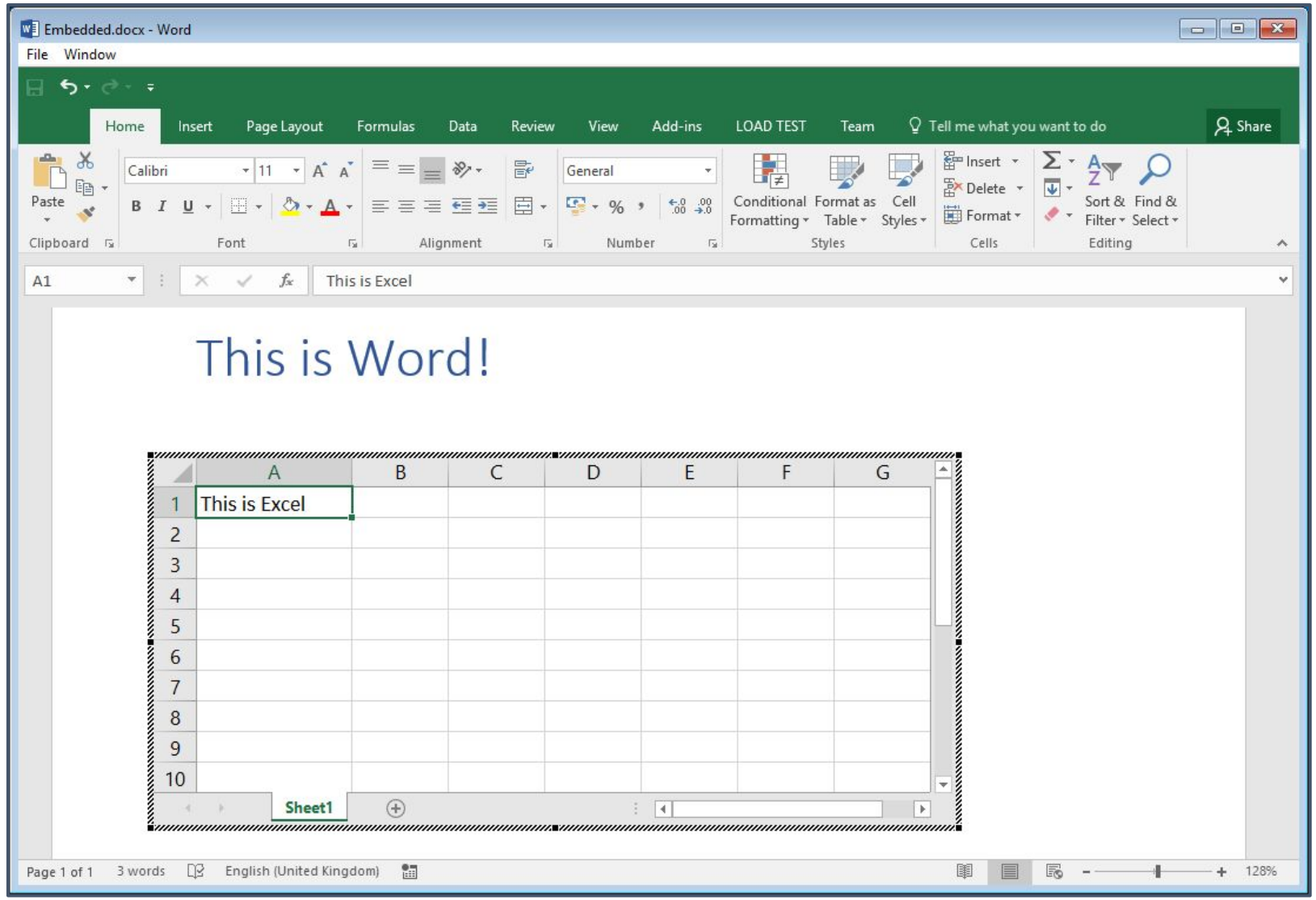
https://github.com/tyranid/oleviewdotnet

https://pixabay.com/en/ford-mustang-engine-hood-bonnet-1242192/

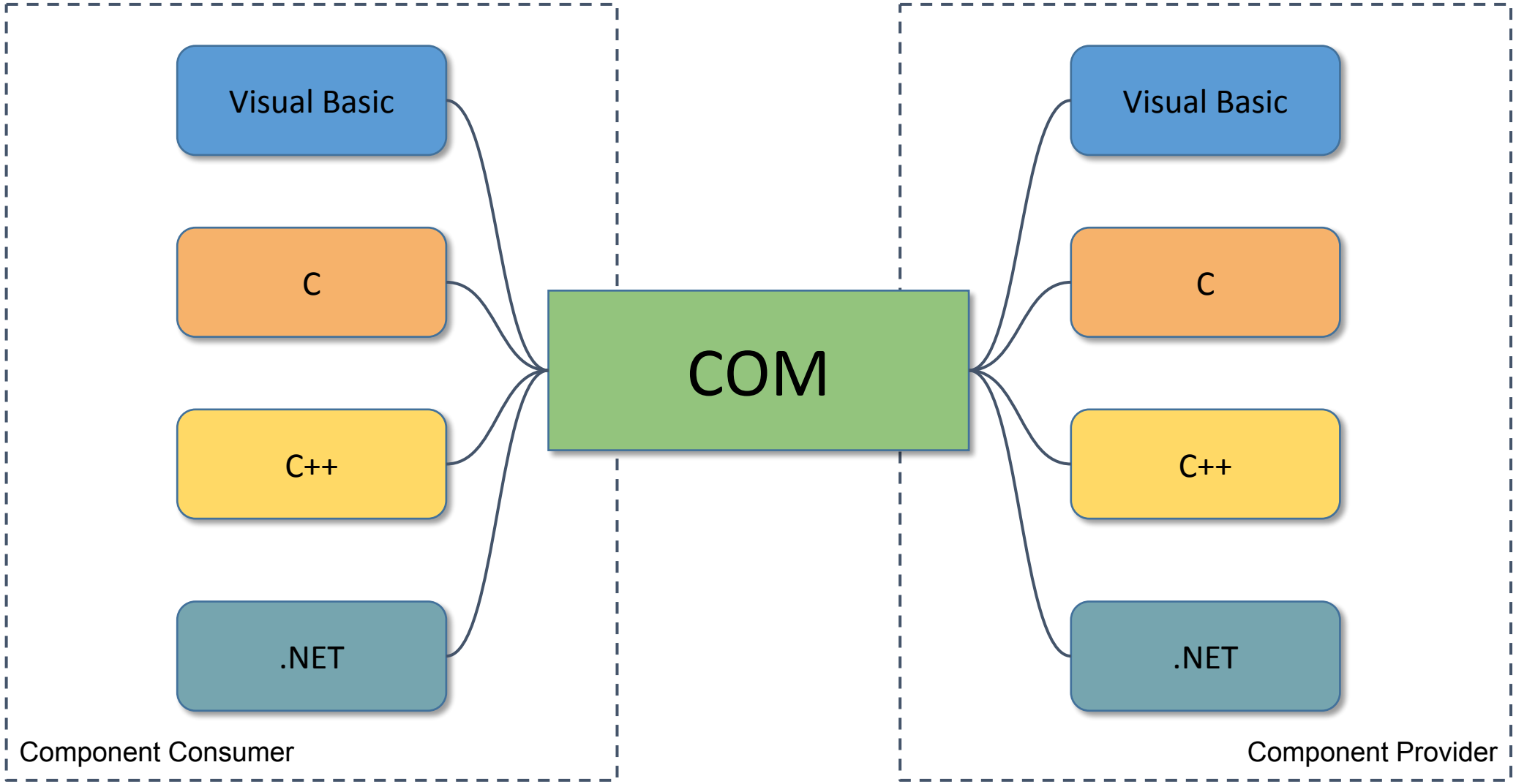
“Any sufficiently complex middleware is indistinguishable from magic.”

Arthur C. Clarke’s Third Law of Software Development

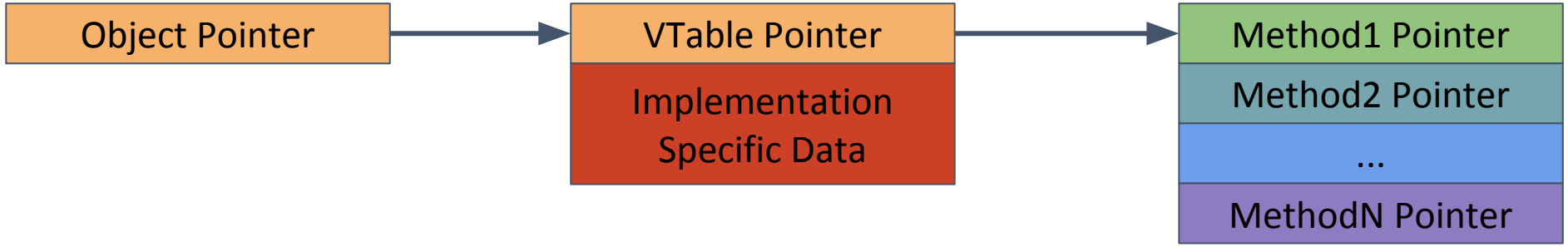
In the Beginning was OLE



Interoperability Heaven



Common ABI




```
struct ObjectVTable {
  void (*SetInt)(struct Object* This, int i);
  int  (*GetInt)(struct Object* This);
};

struct Object {
  struct ObjectVTable* Vtbl;
  // Implementation specific data follows.
};

struct Object* obj;
obj->Vtbl->SetInt(obj, 1234);
```

Object pointer is first.
Arguments passed left to right

VTable Pointer at start

The Casting Problem

```
struct Interface1 {
  virtual void A() = 0;
};

struct Interface2 {
  virtual void B() = 0;
};

class Object : public Interface1,
               public Interface2 {
  void A() {}
  void B() {}
};
```

Define a pure virtual "Interface"

Derive from Interface and implement

```
// Okay (mostly).
Interface1* intf1 = new Object;
// Incredibly bad idea.
Interface2* intf2 = (Interface2*)intf1;
```

No no no!!!

IUnknown, the Root of all COM Evil

```
DEFINE_GUID(IID_IUnknown, 00000000-0000-0000-C000-000000000046");
struct IUnknown {
  HRESULT QueryInterface(GUID& iid, void** ppv);
  LONG AddRef();
  LONG Release();
};
```

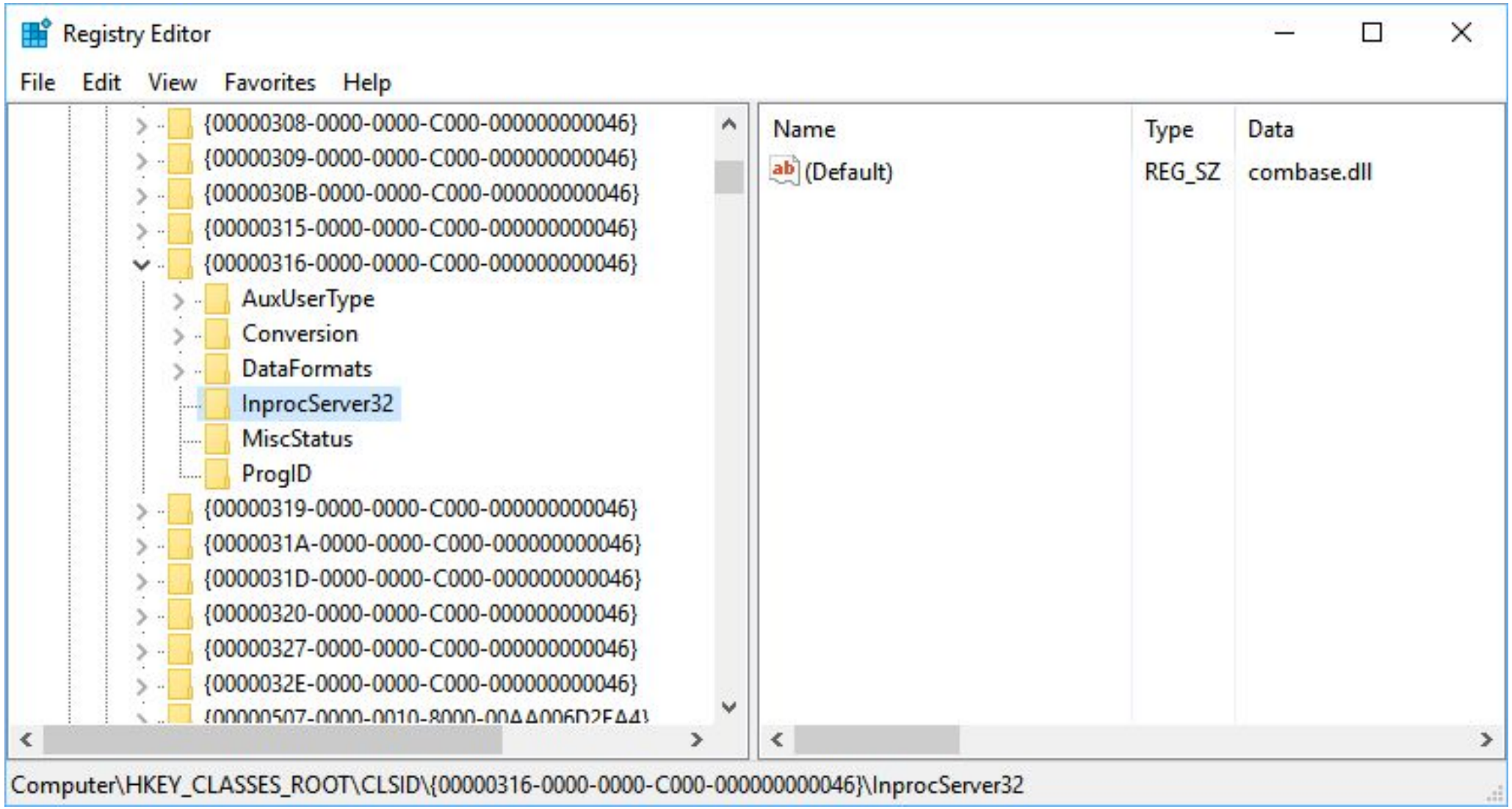
Used to reference count the object.

Interfaces defined using a 128 bit Globally Unique ID.

```
struct Interface1 : public IUnknown {};
struct Interface2 : public IUnknown {};

Interface2* intf2;
if (intf1->QueryInterface(IID_Interface2,
                          (void**)&intf2) >= 0) {
  // Success, we can call methods.
  intf2->Release();
}
```

Cast is a GIANT code smell!
¯_(ツ)_/¯

Class Registration



Class Factories

```
DEFINE_GUID(IID_ClassFactory,
    "00000001-0000-0000-C000-000000000046");

struct IClassFactory : public IUnknown {
  HRESULT CreateInstance(
    IUnknown *pUnkOuter,
    REFIID riid,
    void **ppvObject);

  HRESULT LockServer(BOOL fLock);
};
```

Creating Class Factories and Instances

```
HRESULT CoGetClassObject(
  REFCLSID      rclsid,
  DWORD         dwClsContext,
  COSERVERINFO *pServerInfo,
  REFIID        riid,
  LPVOID        *ppv
);
```

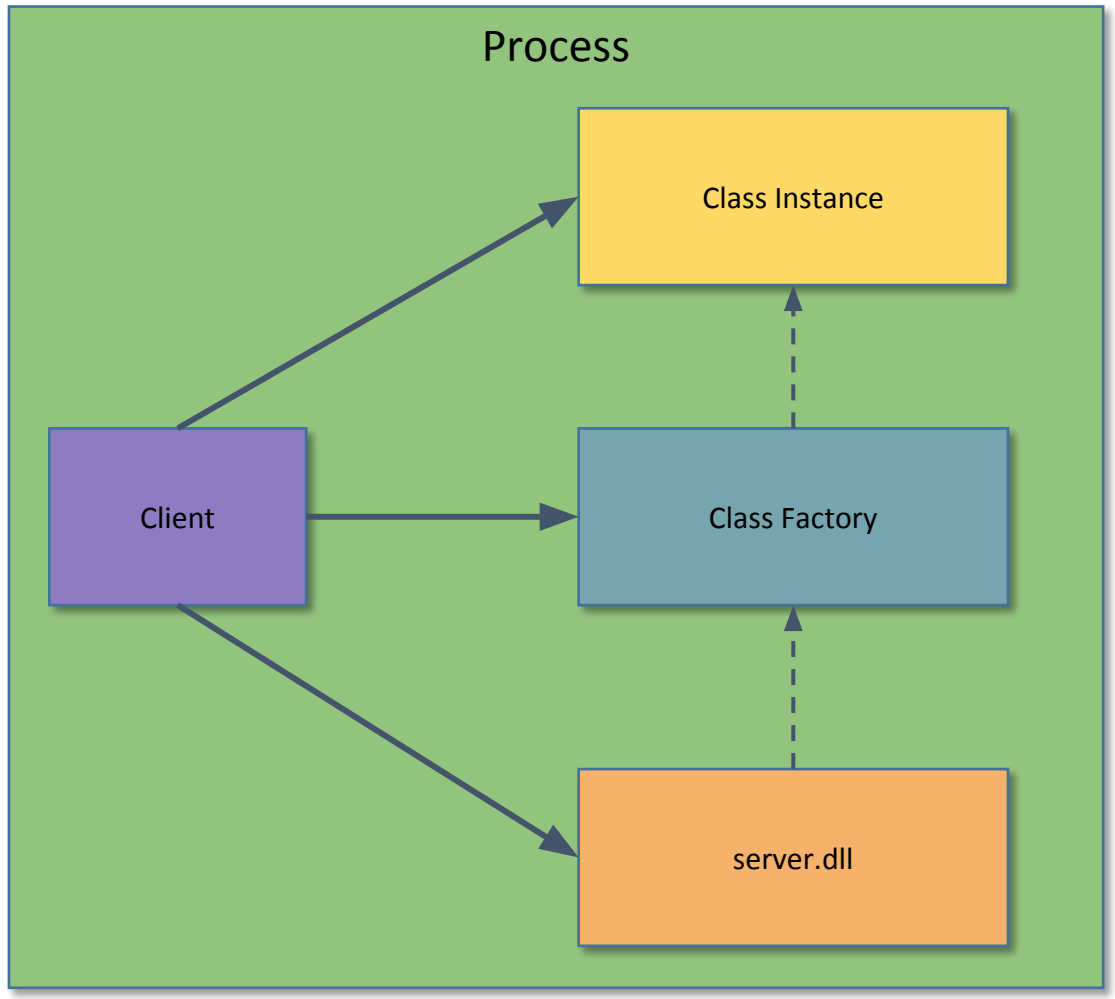
Specifies what type of server to lookup:

- CLSCTX_INPROC_SERVER
- CLSCTX_INPROC_HANDLER
- CLSCTX_LOCAL_SERVER
- CLSCTX_REMOTE_SERVER

```
HRESULT CoCreateInstanceEx(
  REFCLSID      rclsid,
  IUnknown      *punkOuter,
  DWORD         dwClsCtx,
  COSERVERINFO *pServerInfo,
  DWORD         dwCount,
  MULTI_QI      *pResults
);
```

Specify remote server information is required (more on this later).

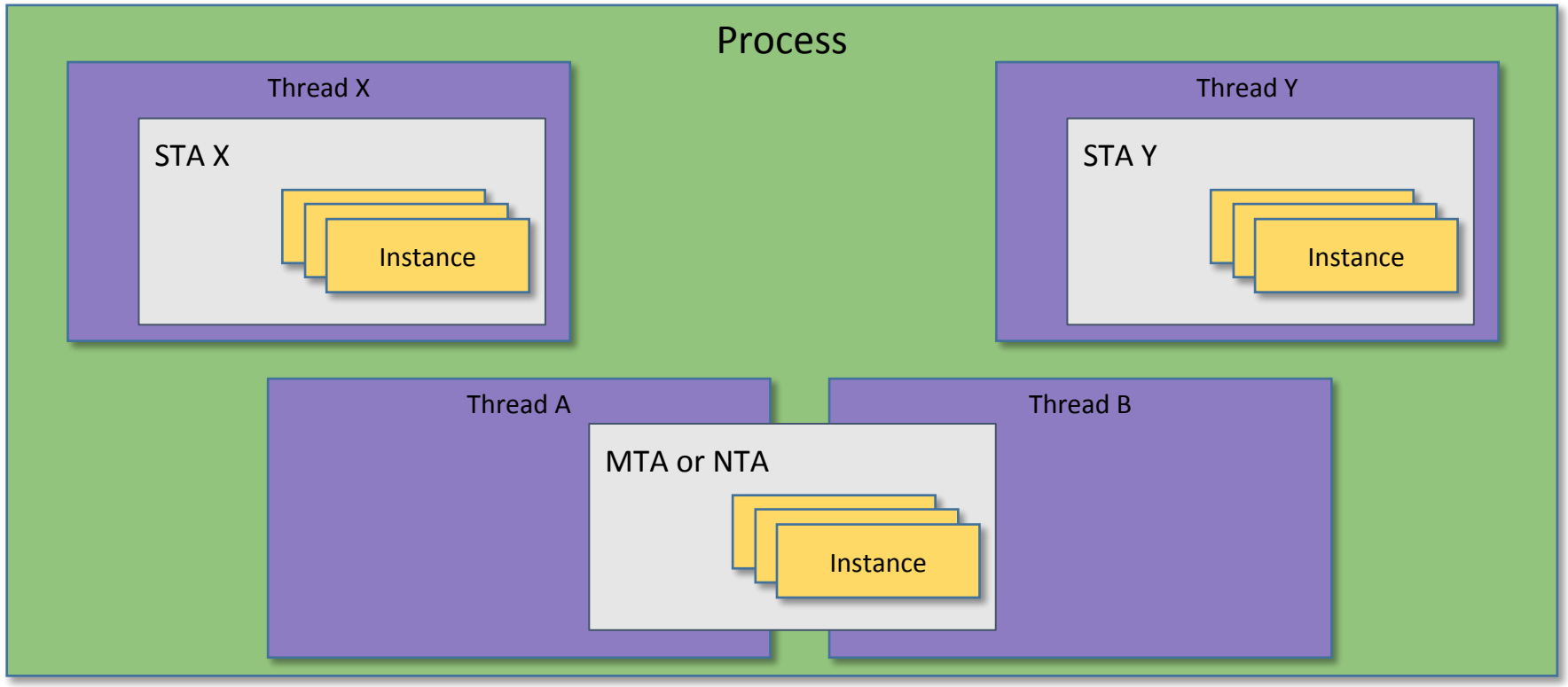
In-Process Server



- DLL server filename specified in *InProcServer32* key.
- DLL loaded into process and class factory created by calling exported method:

```
HRESULT DllGetClassObject(REFCLSID rclsid,
                          REFIID riid,
                          LPVOID *ppv);
```

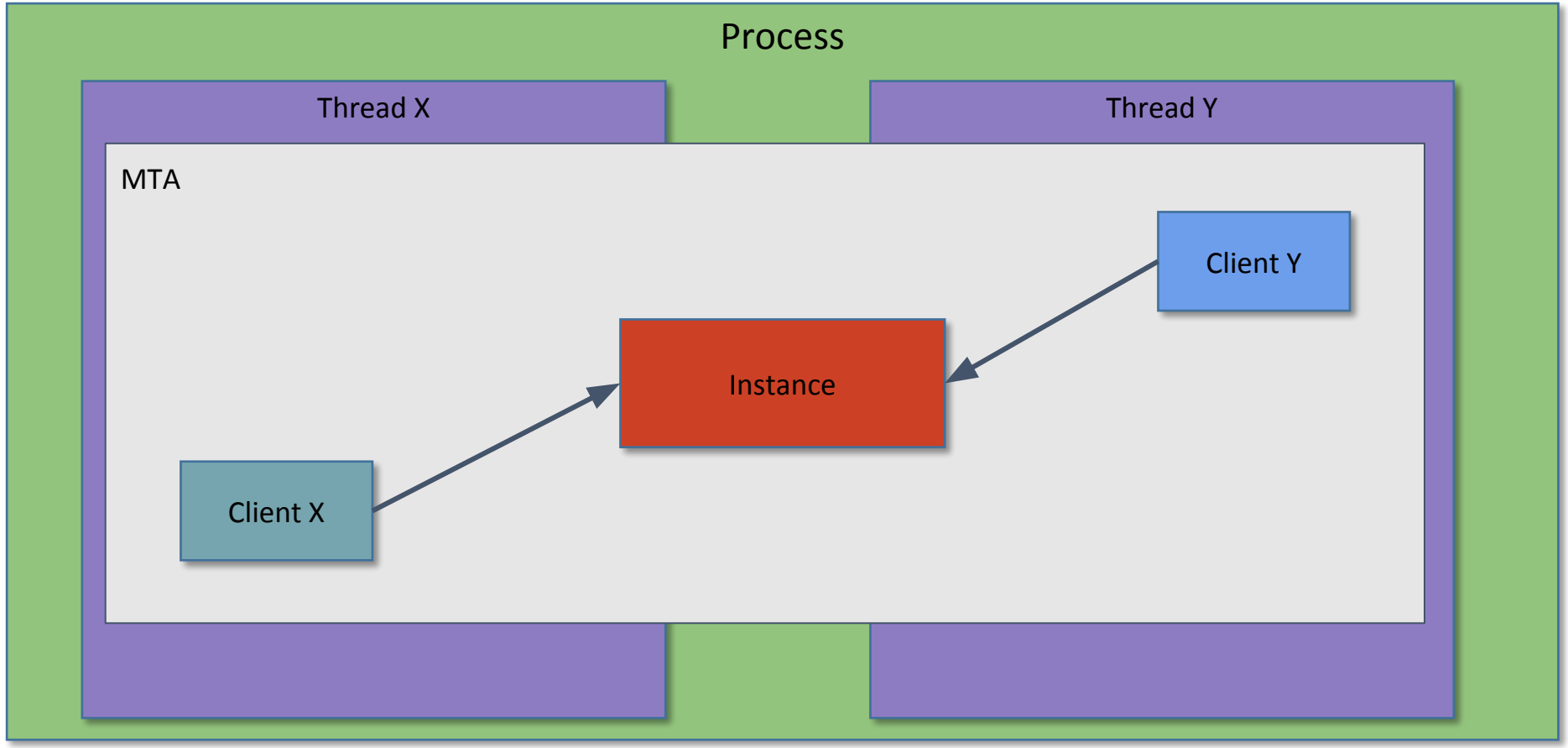
COM Apartments



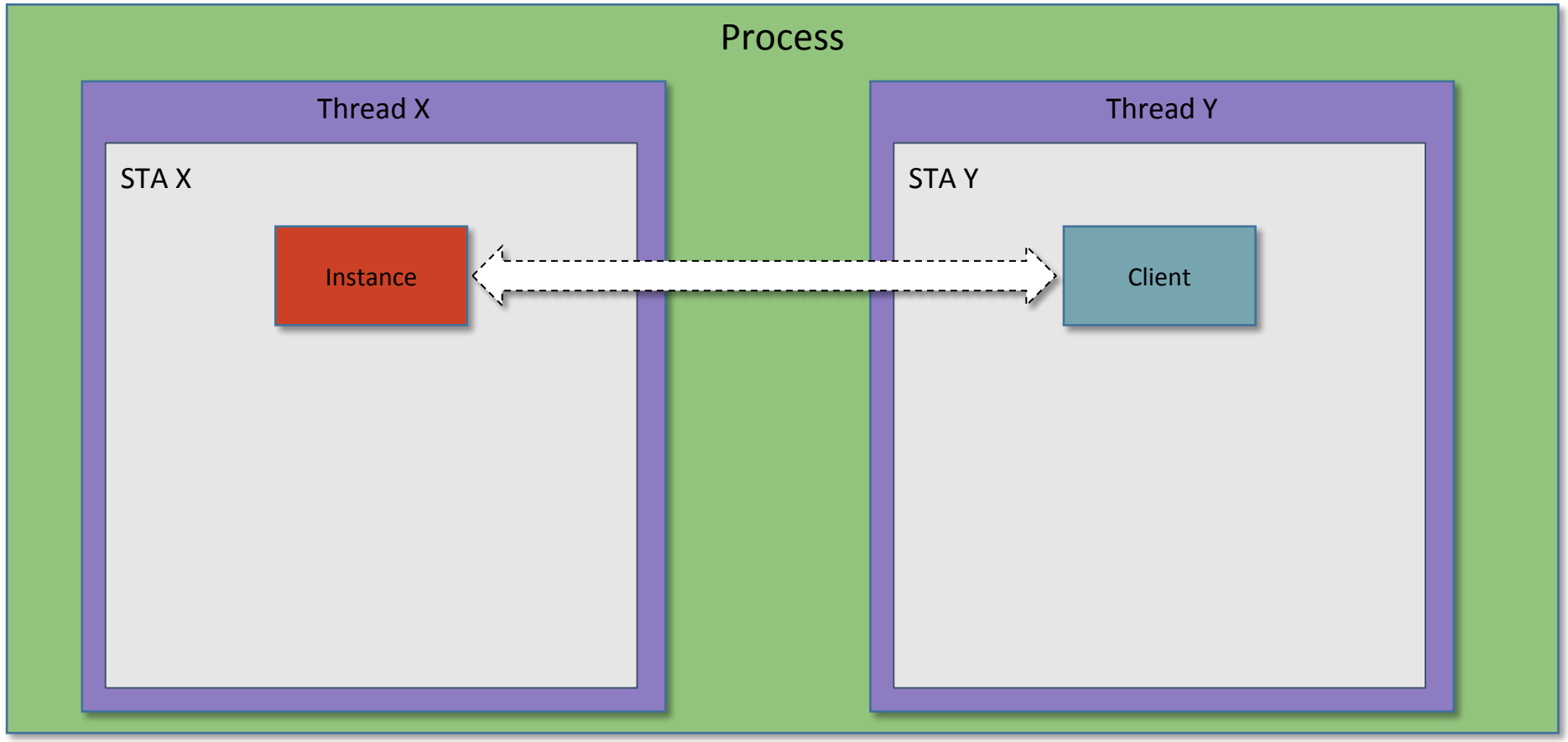

```
HRESULT CoInitializeEx(
  LPVOID pvReserved,
  DWORD  dwCoInit
);
```

```
COINIT_APARTMENTTHREADED
or
COINIT_MULTITHREADED
```

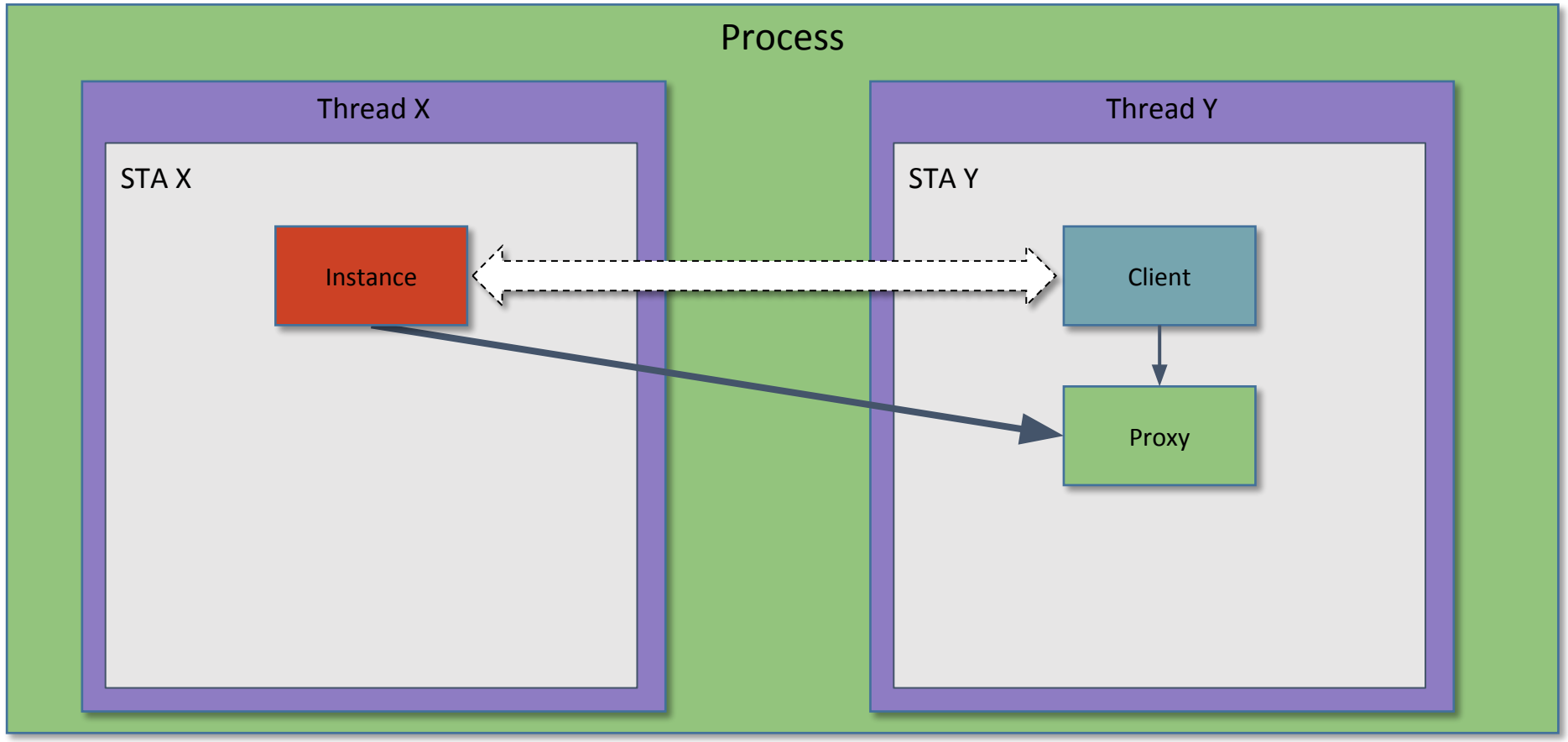
Multi-Threaded Apartments (MTA)



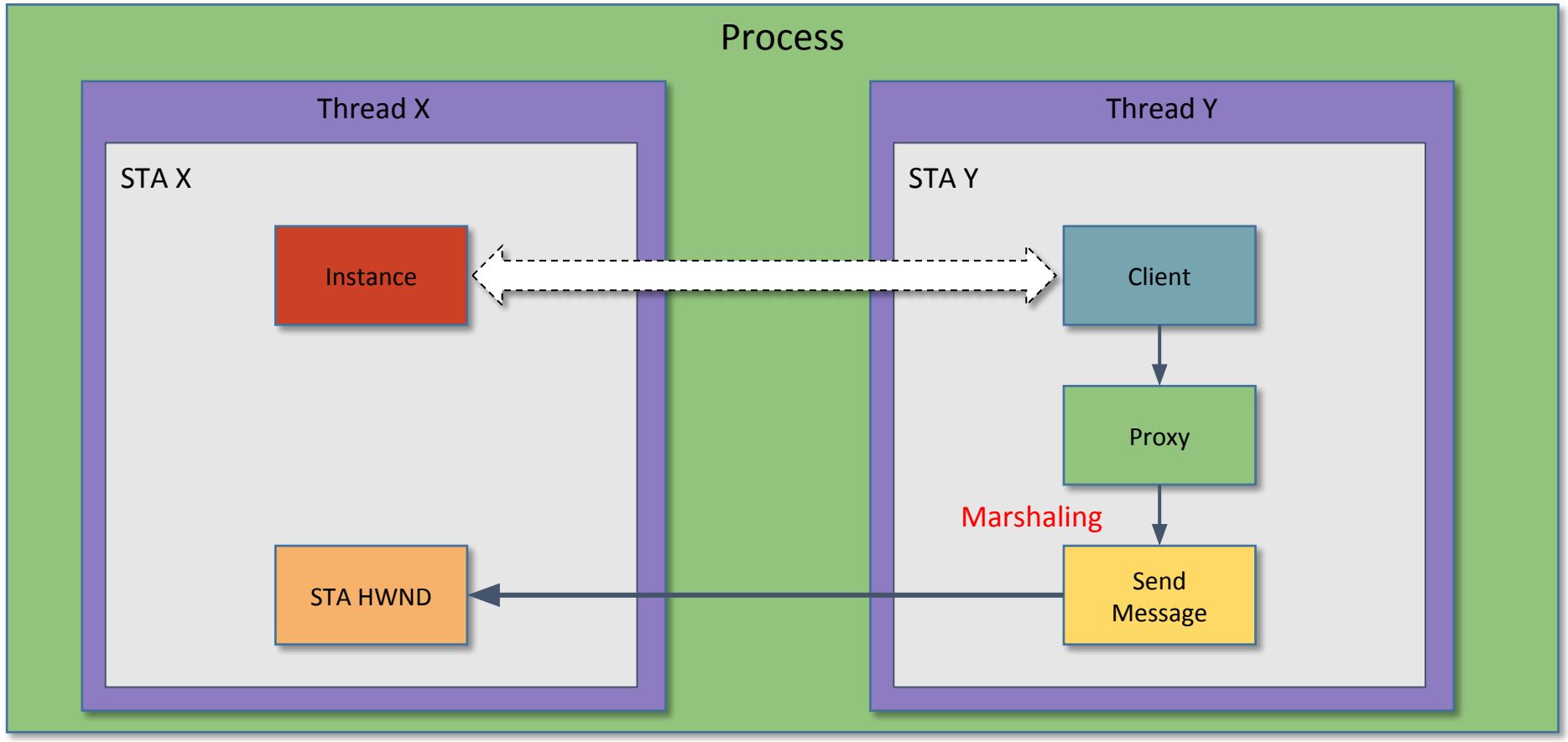
Single Threaded Apartments (STA)



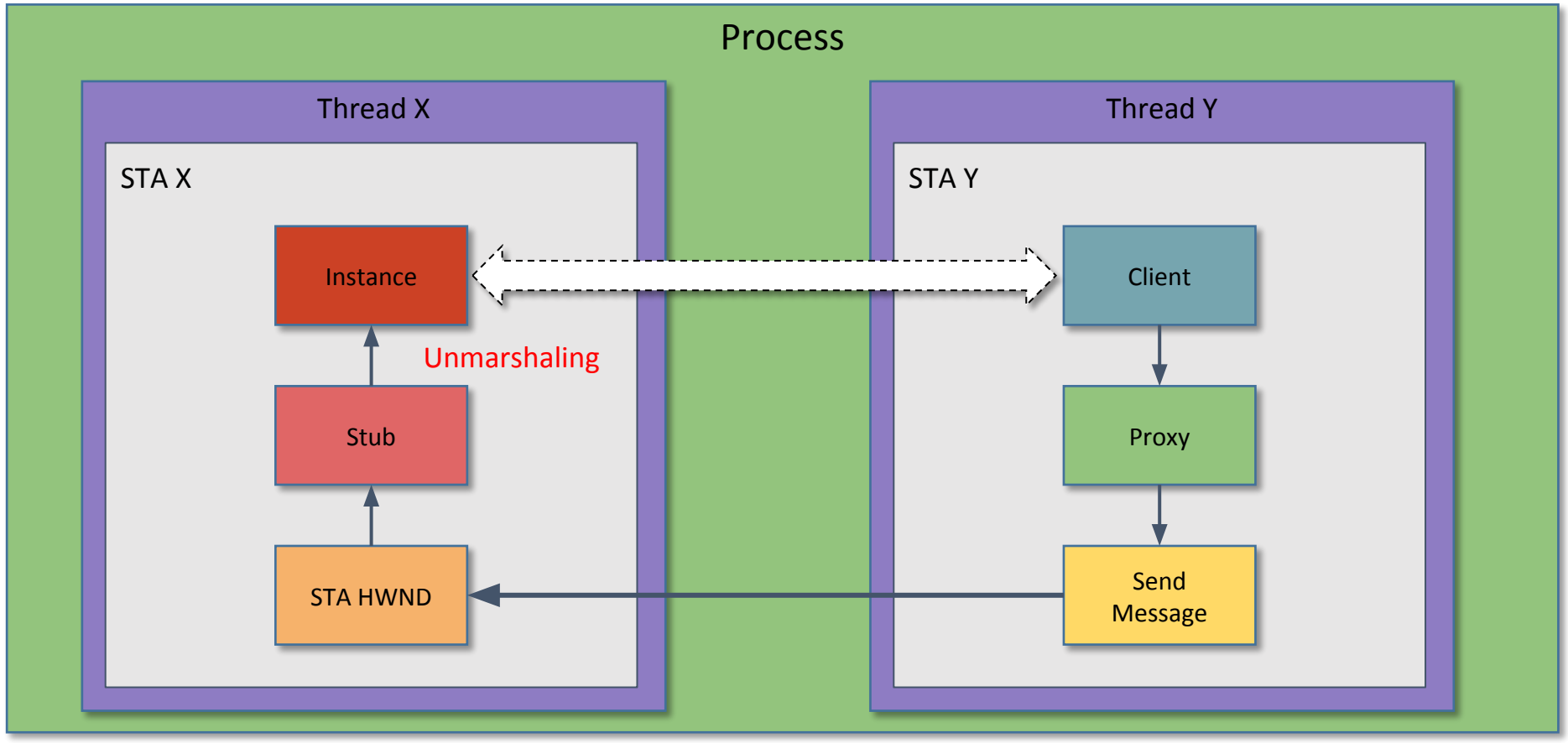
Single Threaded Apartments (STA)



Single Threaded Apartments (STA)

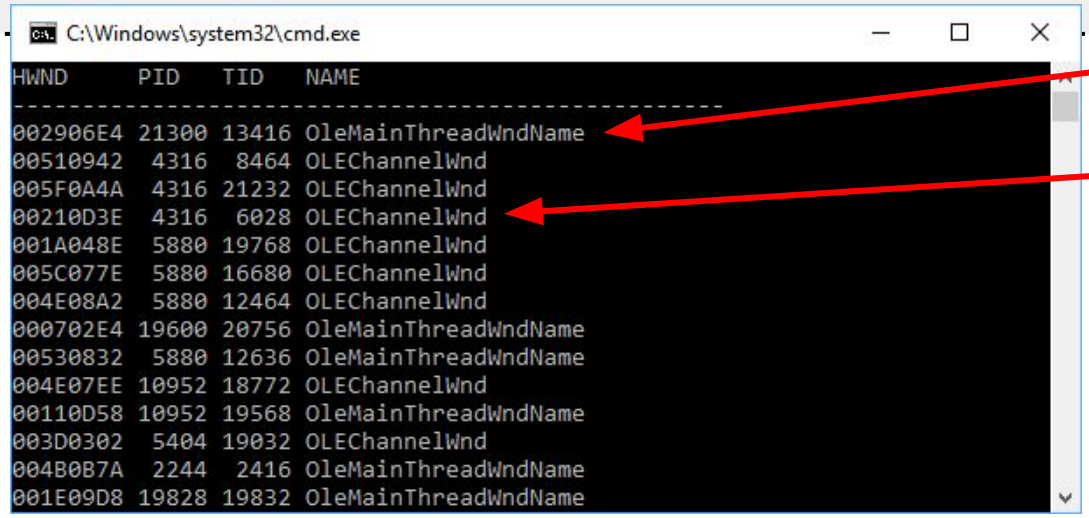


Single Threaded Apartments (STA)



The Mystery Window

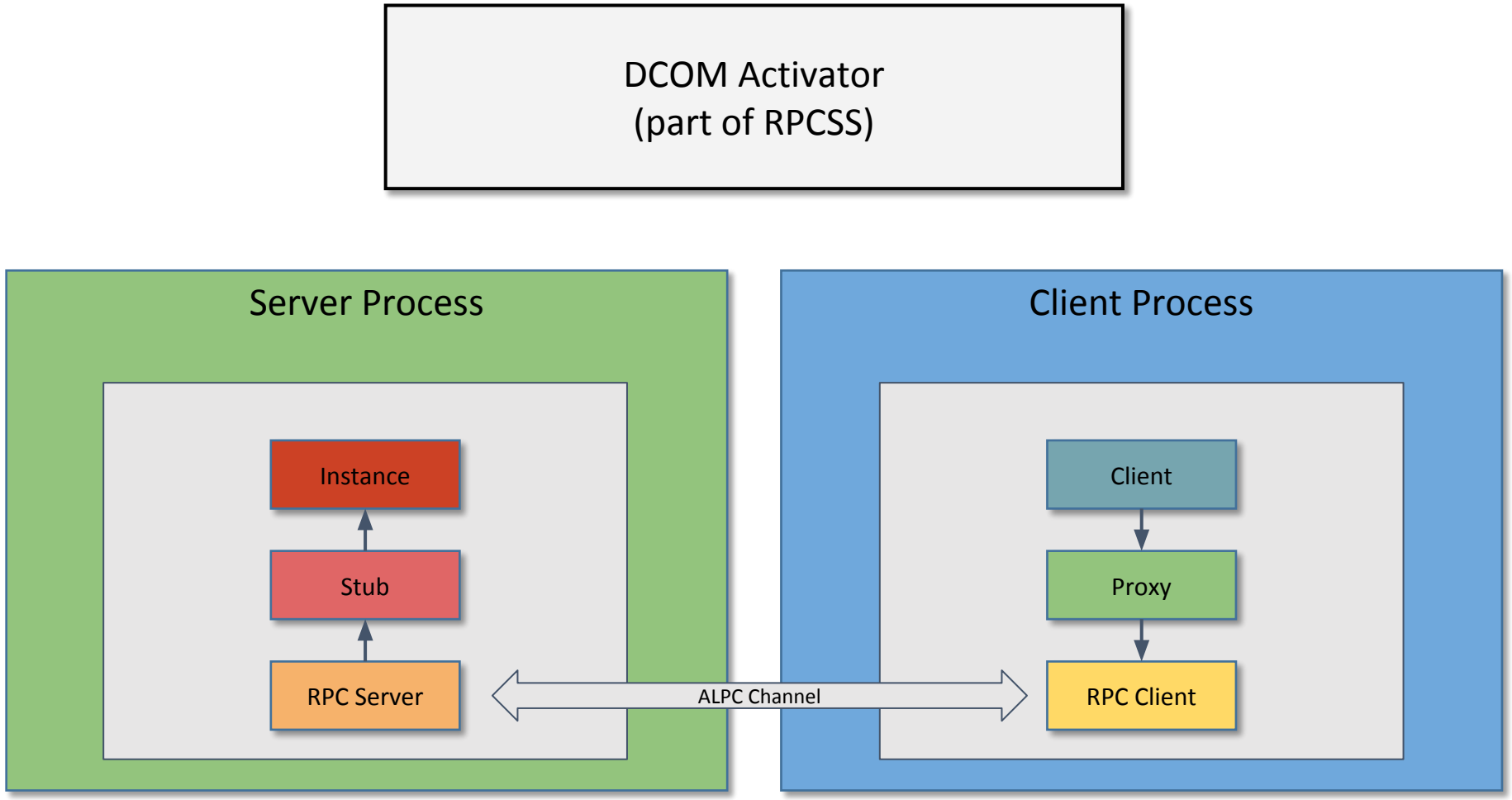
```
HWND hwnd = FindWindowEx(HWND_MESSAGE, NULL, NULL, NULL);
while (hwnd) {
  WCHAR name[256] = {};
  if (GetWindowText(hwnd, name, _countof(name))
      && _wcsnicmp(name, L"ole", 3) == 0) {
    DWORD pid = 0;
    DWORD tid = GetWindowThreadProcessId(hwnd, &pid);
    printf("%p %5d %5d %ls\n", hwnd, pid, tid, name);
  }
  hwnd = GetNextWindow(hwnd, GW_HWNDNEXT);
}
```



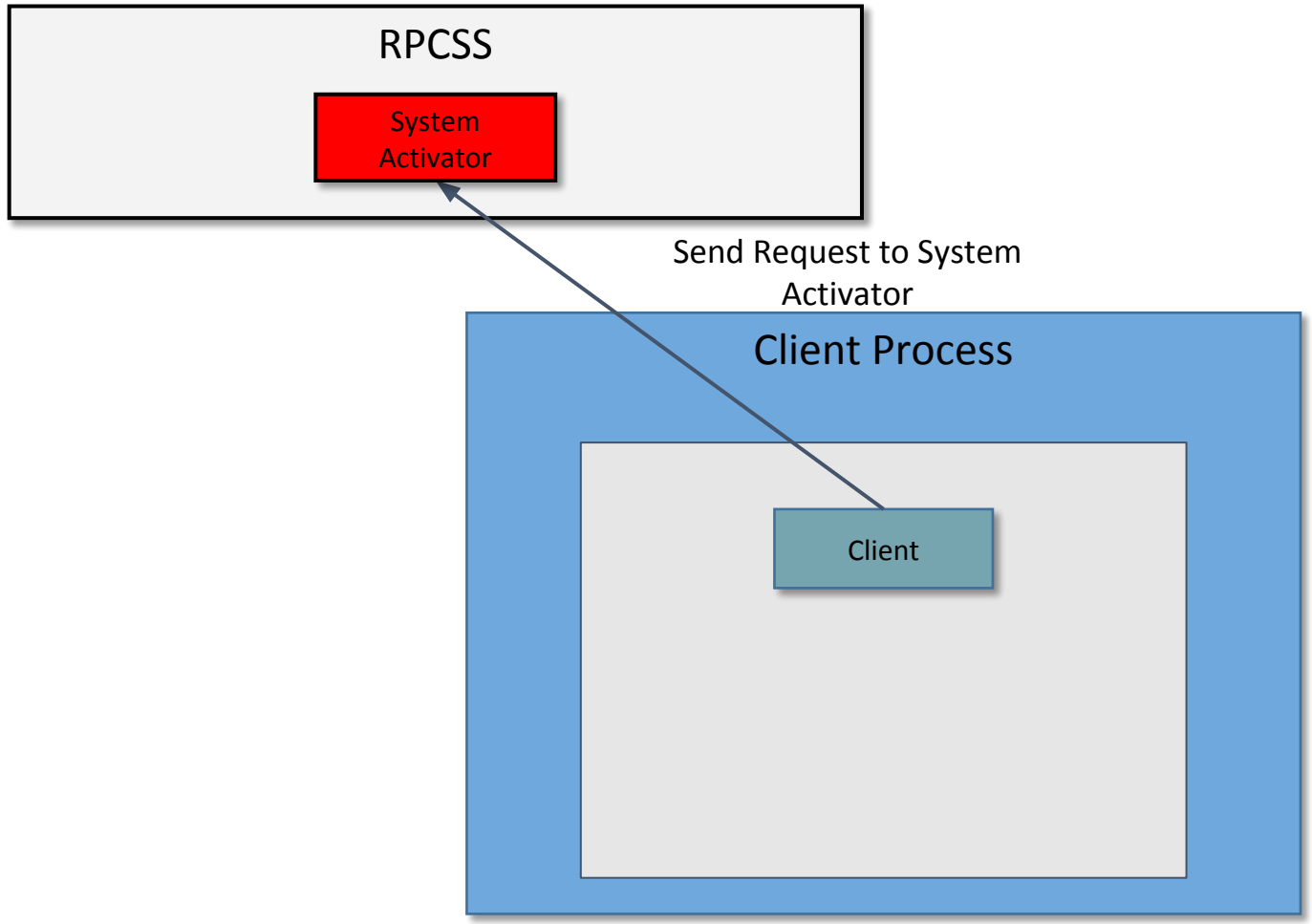
"Main" STA window.

Per-Thread STA window.

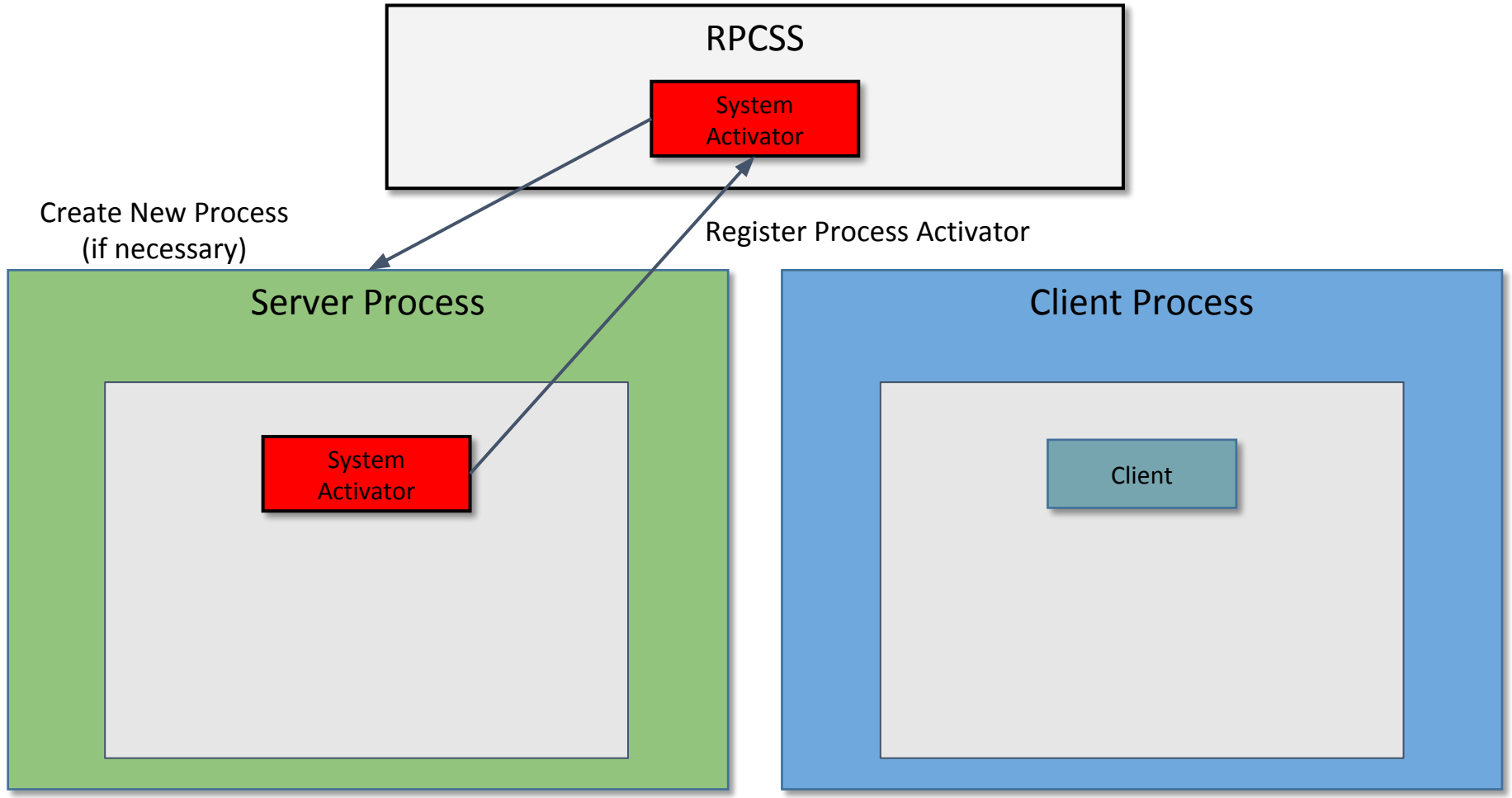
Local Server Activation



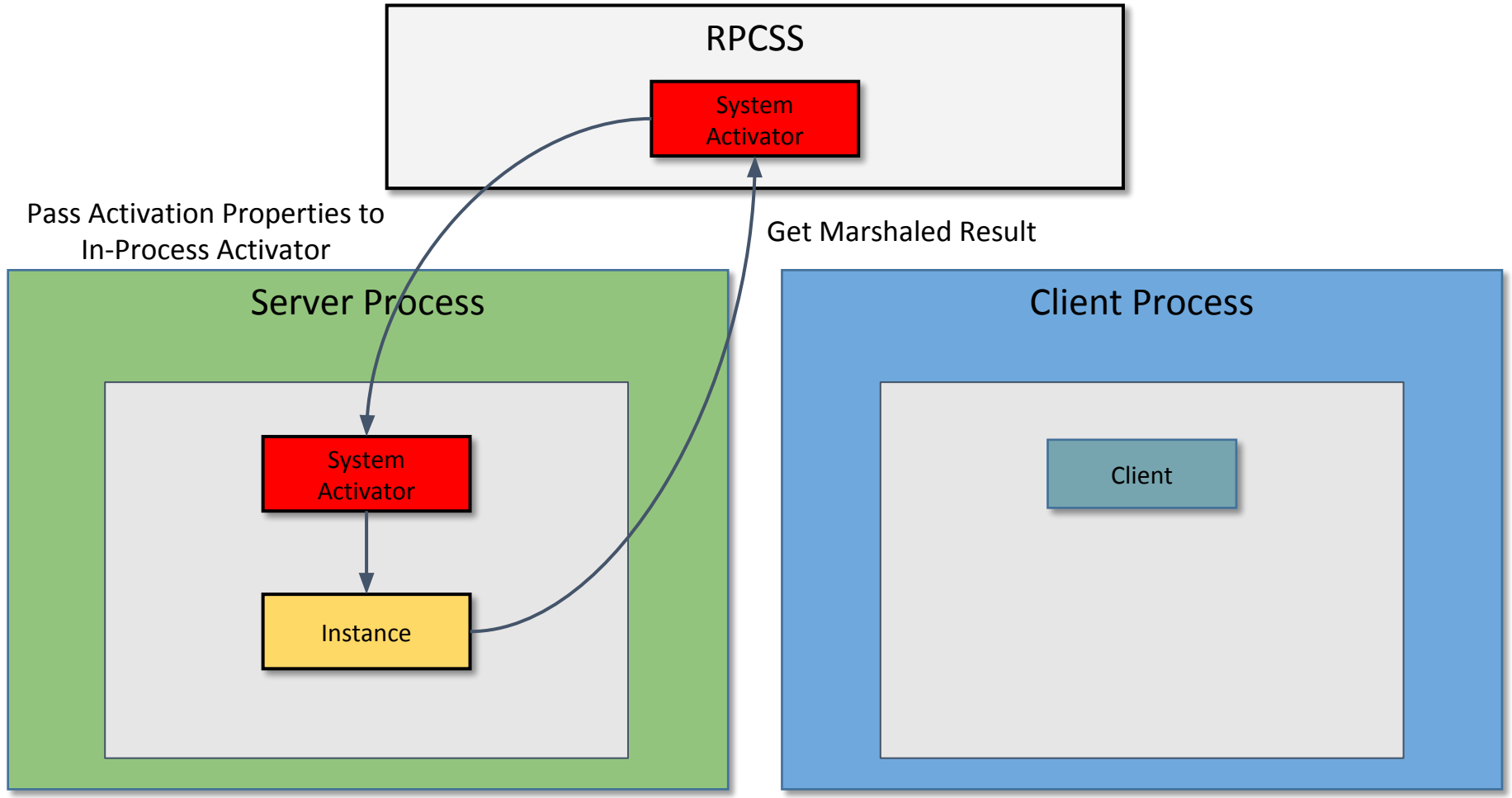
Local Server Activation



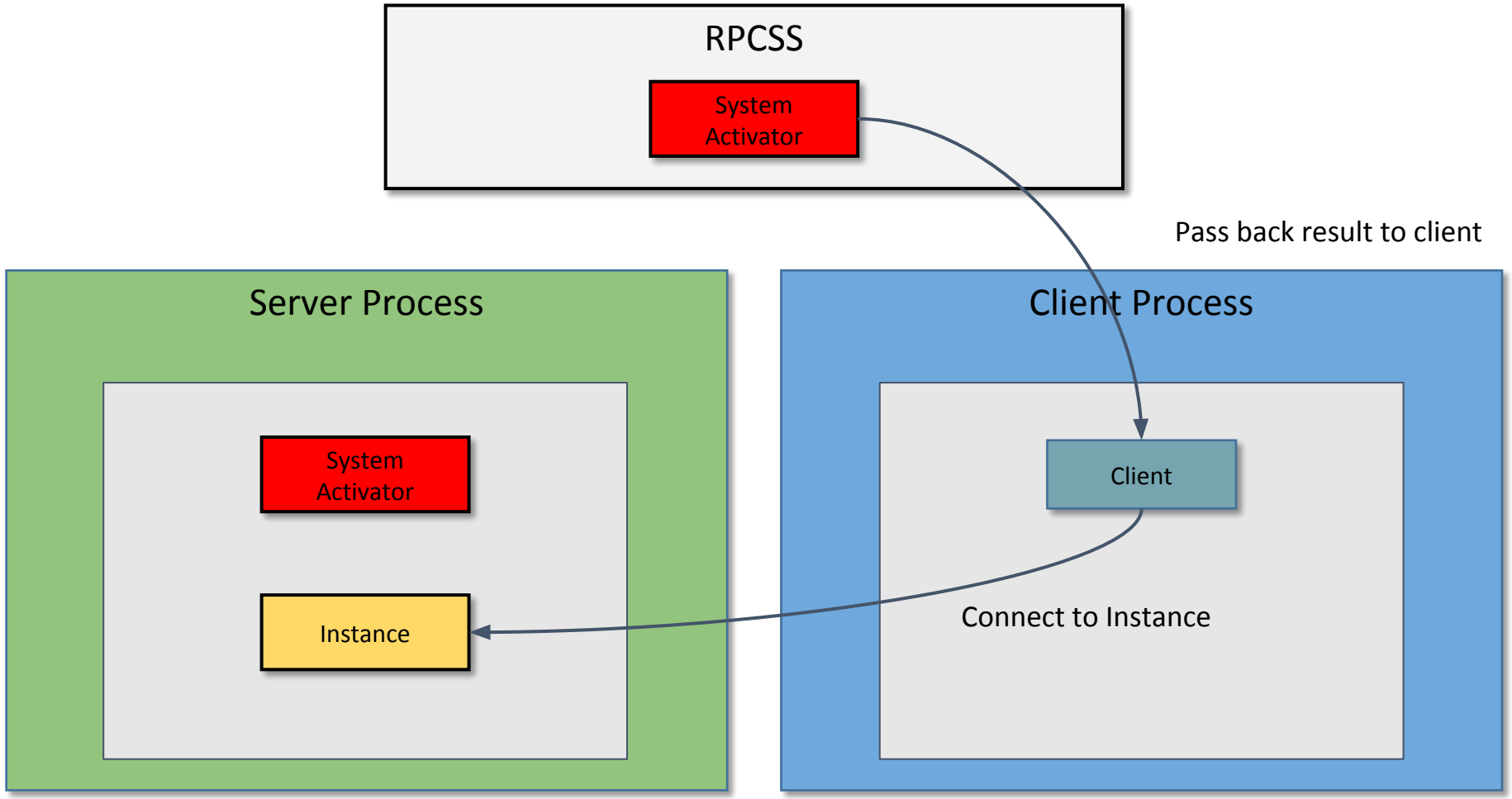
Local Server Activation



Local Server Activation



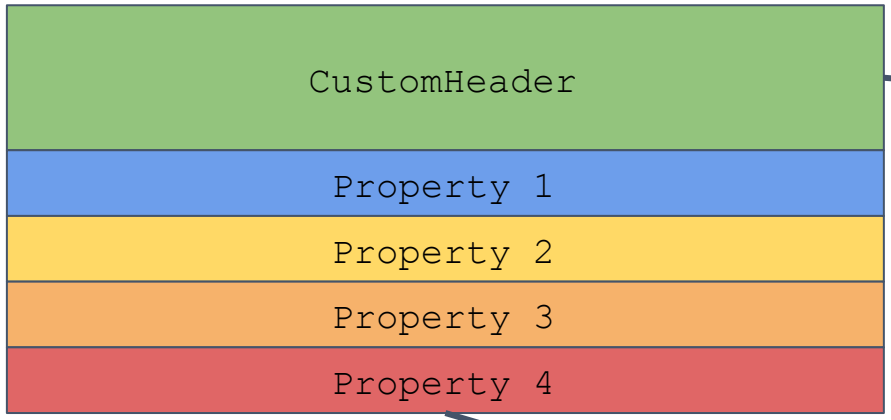
Local Server Activation



System Activator

```
DEFINE_GUID(IID_ISystemActivator,
            "000001a0-0000-0000-c000-000000000046")
struct ISystemActivator : public IUnknown {
  HRESULT GetClassObject(
        IActivationPropertiesIn *pActProperties,
        IActivationPropertiesOut **ppActProperties);

  HRESULT CreateInstance(
        IUnknown *pUnkOuter,
        IActivationPropertiesIn *pActProperties,
        IActivationPropertiesOut **ppActProperties);
};
```

Activation Properties In




```
struct CustomHeader {
   DWORD totalSize;
   DWORD headerSize;
   DWORD dwReserved;
   DWORD destCtx;
   DWORD cIfs;
   CLSID classInfoClsid;
   CLSID *pclsid;
   DWORD *pSizes;
   CustomOpaqueData  *opaqueData;
};
```

List of GUIDs and Sizes of following Property Blobs

```
struct InstantiationInfoData {
   CLSID classId;
   DWORD classCtx;
   DWORD actvflags;
   long fIsSurrogate;
   DWORD cIID;
   DWORD instFlag;
   IID *pIID;
   DWORD thisSize;
   COMVERSION clientCOMVersion;
};
```

CLSID to create.

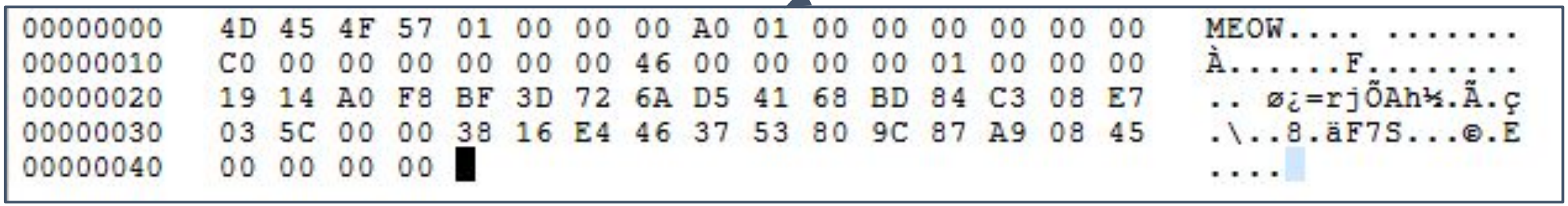
```
enum ACTIVATION_FLAGS {
   ACTVFLAGS_DISABLE_AAA,
   ACTVFLAGS_ACTIVATE_32_BIT_SERVER,
   ACTVFLAGS_ACTIVATE_64_BIT_SERVER,
   ACTVFLAGS_NO_FAILURE_LOG,
   ACTVFLAGS_WINRT_LOCAL_SERVER,
   ACTVFLAGS_WINRT_PER_USER_OK,
   ACTVFLAGS_APPCONTAINER,
};
```

List of IIDs to query for.

Activation Properties Out

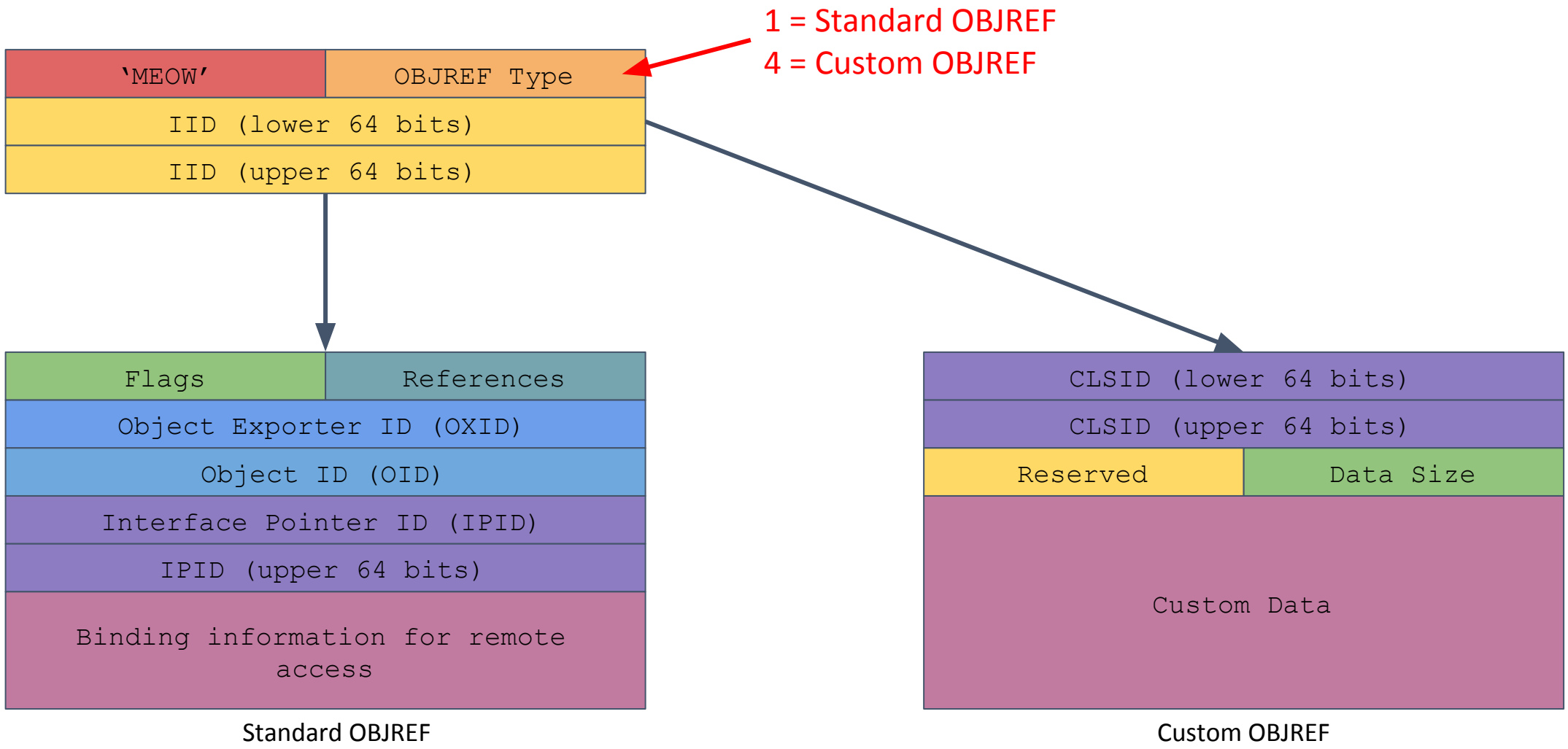
```
struct PropsOutInfo {
   DWORD cIfs;
   IID *piid;
   HRESULT *phresults;
   MInterfacePointer **ppIntfData;
};
```

```
struct MInterfacePointer {
    unsigned long ulCntData;
    byte abData[];
};
```



```
00000000   4D 45 4F 57 01 00 00 00 A0 01 00 00 00 00 00 00   MEOW.... ........
00000010   C0 00 00 00 00 00 00 46 00 00 00 00 01 00 00 00   À......F........
00000020   19 14 A0 F8 BF 3D 72 6A D5 41 68 BD 84 C3 08 E7   .. ø¿=rjÕAh½.Ã.ç
00000030   03 5C 00 00 38 16 E4 46 37 53 80 9C 87 A9 08 45   .\..8.äF7S...©.E
00000040   00 00 00 00                                       ....
```

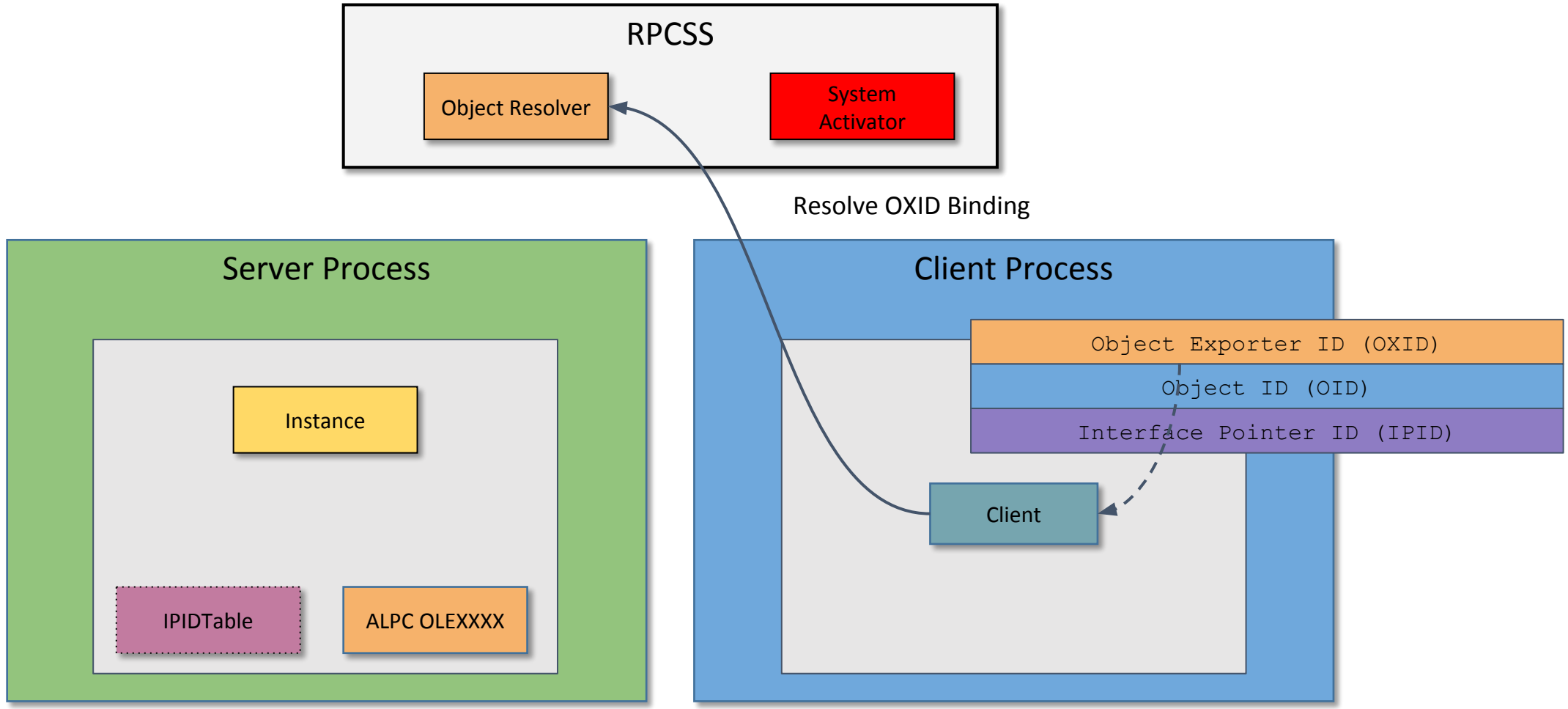
It’s like Marshaling Cats



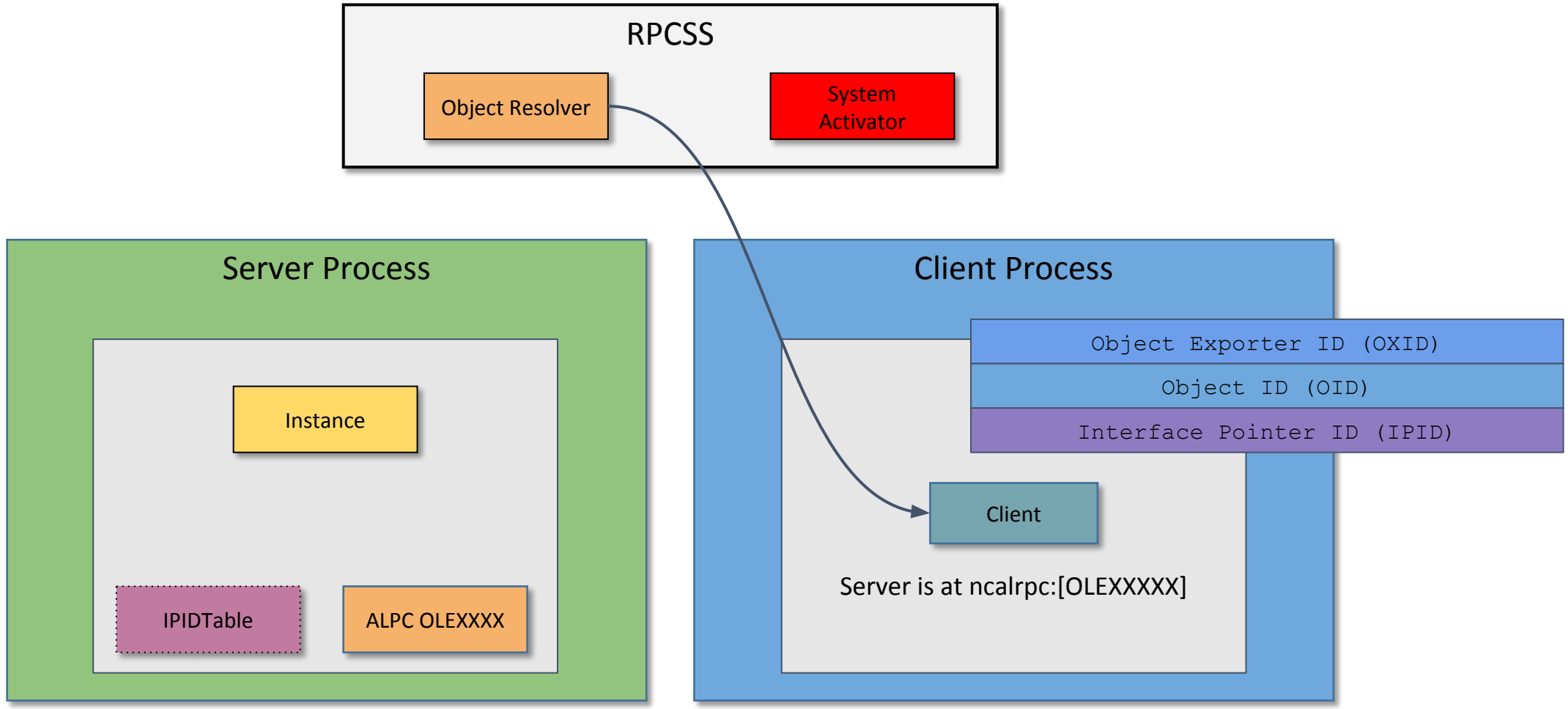
Standard OBJREF

Custom OBJREF

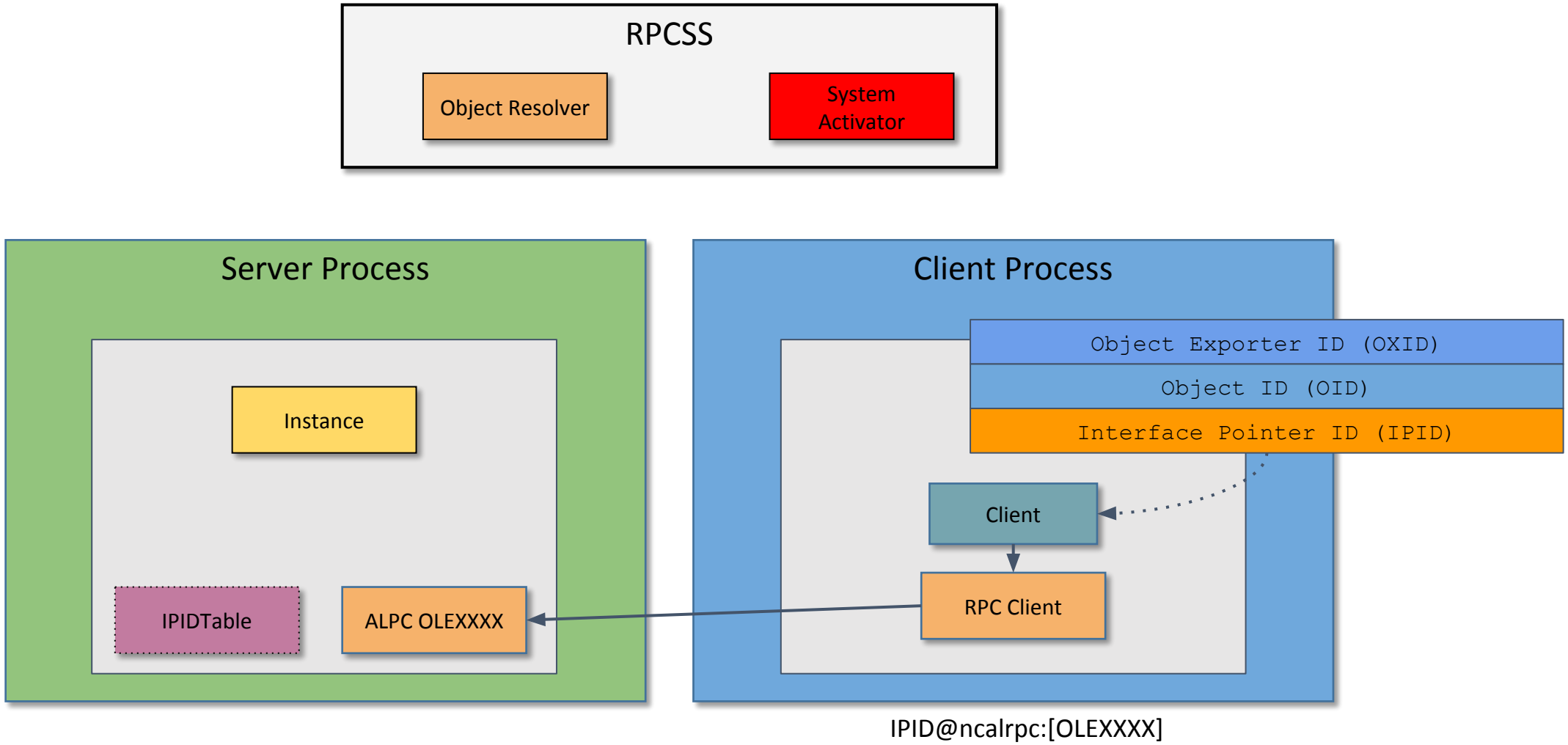
Standard Unmarshaling



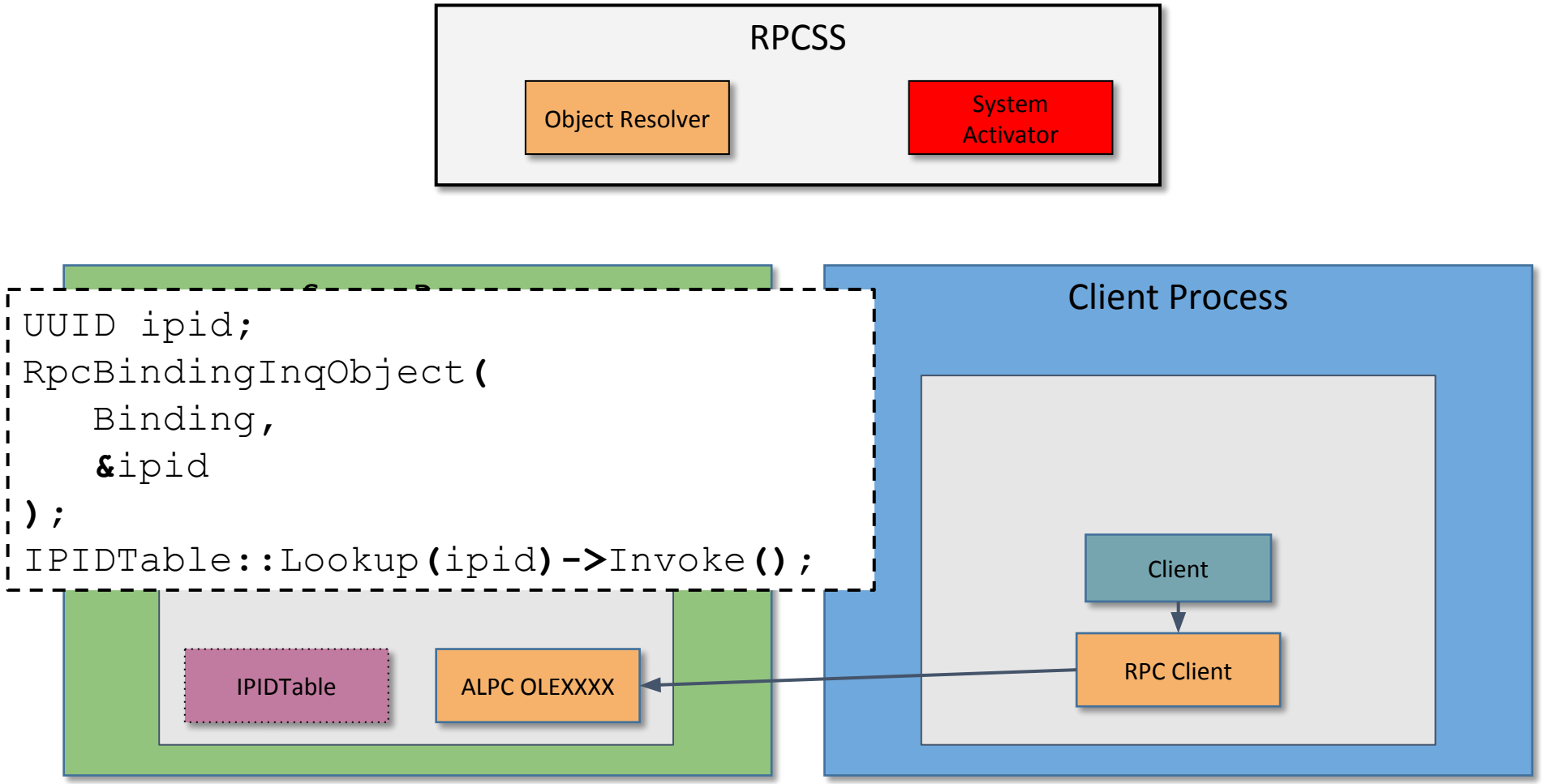
Standard Unmarshaling



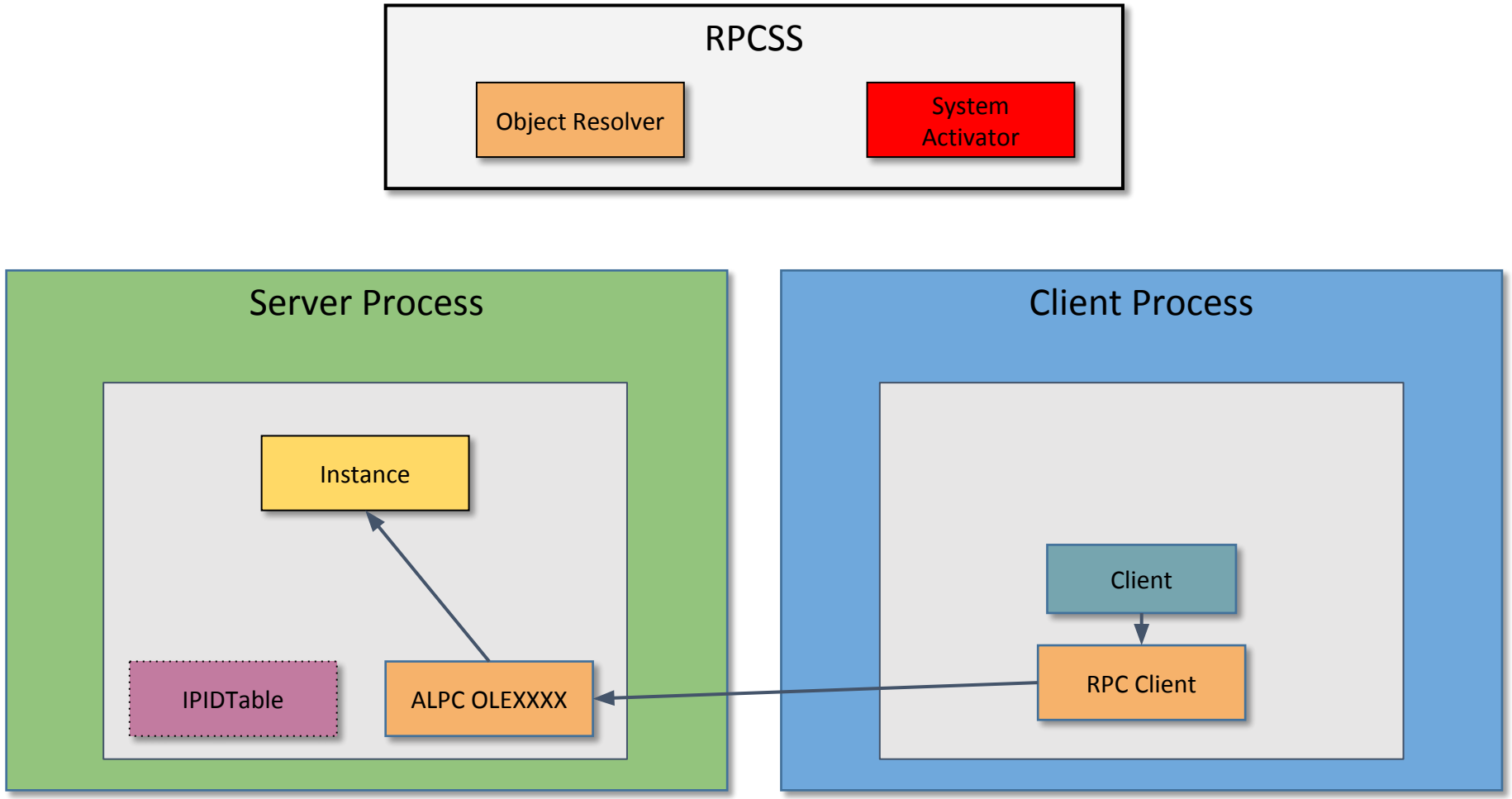
Standard Unmarshaling



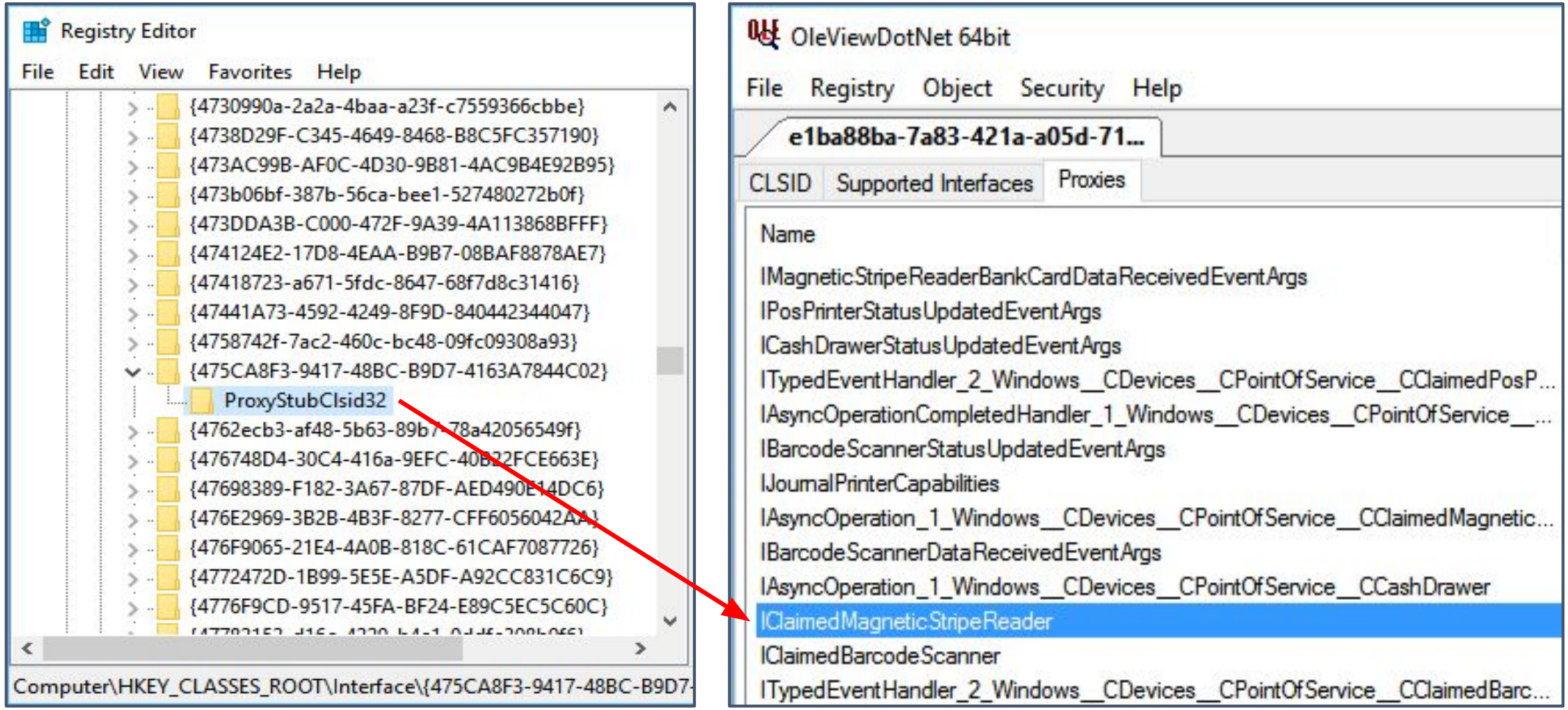
Standard Unmarshaling



Standard Unmarshaling



Interface Proxies and Stubs



Interface Proxy-Stub Factory

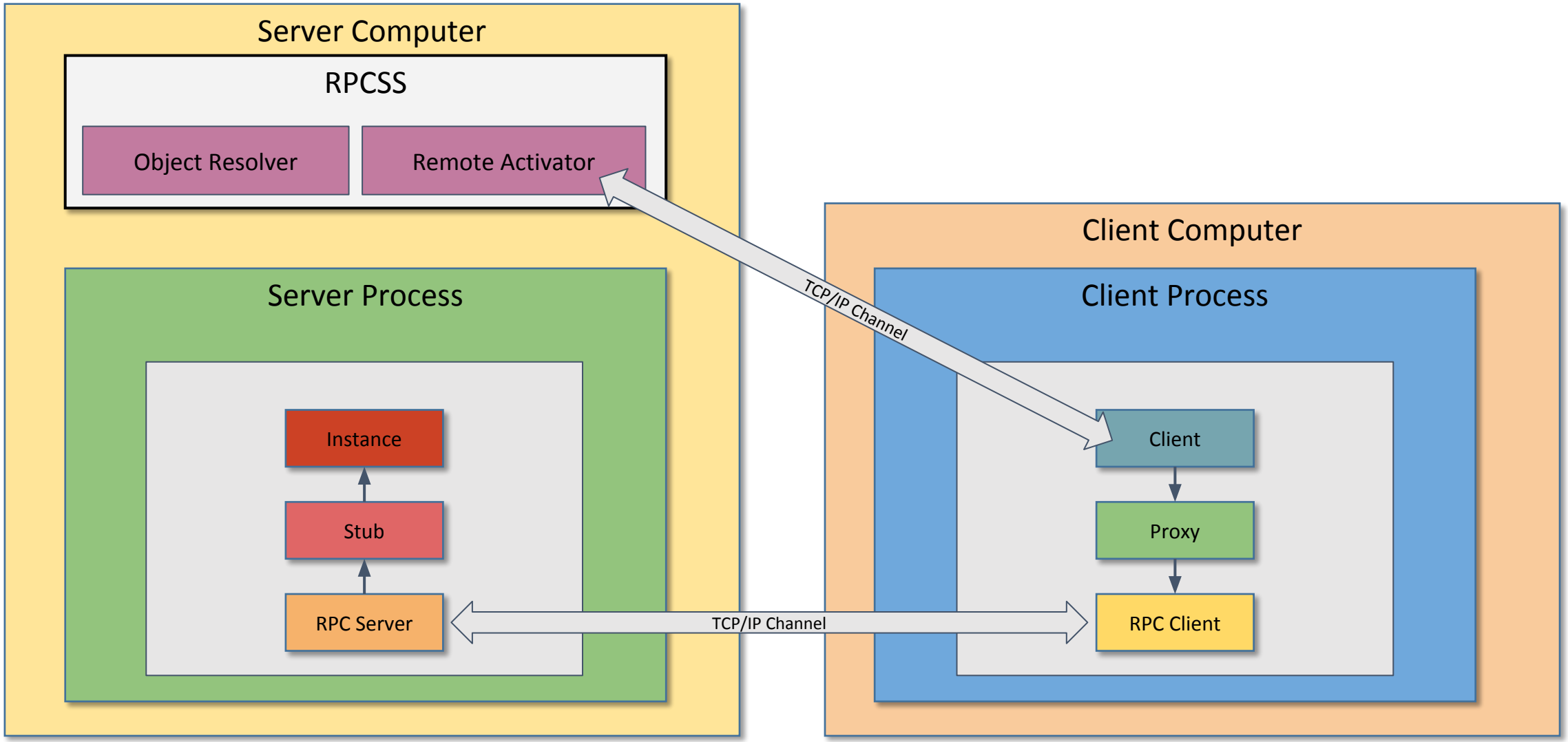
```
DEFINE_GUID(IID_IPSFactoryBuffer,
    "D5F569D0-593B-101A-B569-08002B2DBF7A")
struct IPSFactoryBuffer : public IUnknown
{
    HRESULT CreateProxy(
        IUnknown *pUnkOuter,
        REFIID riid,
        IRpcProxyBuffer **ppProxy,
        void **ppv);

    HRESULT CreateStub(
        REFIID riid,
        IUnknown *pUnkServer,
        IRpcStubBuffer **ppStub);
};
```

IRemUnknown

```
struct customREMOTE_REPLY_SCM_INFO {
  OXID               Oxid;
  DUALSTRINGARRAY   *pdsaOxidBindings;
  IPID               ipidRemUnknown;
  DWORD              authnHint;
  COMVERSION         serverVersion;
};
```

```
DEFINE_GUID(IID_IRemUnknown,
    "00000131-0000-0000-C000-000000000046")
struct IRemUnknown : public IUnknown {
    HRESULT RemQueryInterface(
        REFIPID ripid,
        unsigned long cRefs,
        unsigned short cIids,
        IID *iids,
        PREMQIRESULT *ppQIResults);

    HRESULT RemAddRef(
       unsigned short cInterfaceRefs,
       REMINTERFACEREF InterfaceRefs[],
       HRESULT *pResults);

    HRESULT RemRelease(
        unsigned short cInterfaceRefs,
        REMINTERFACEREF InterfaceRefs[]);
};
```

Remote Server



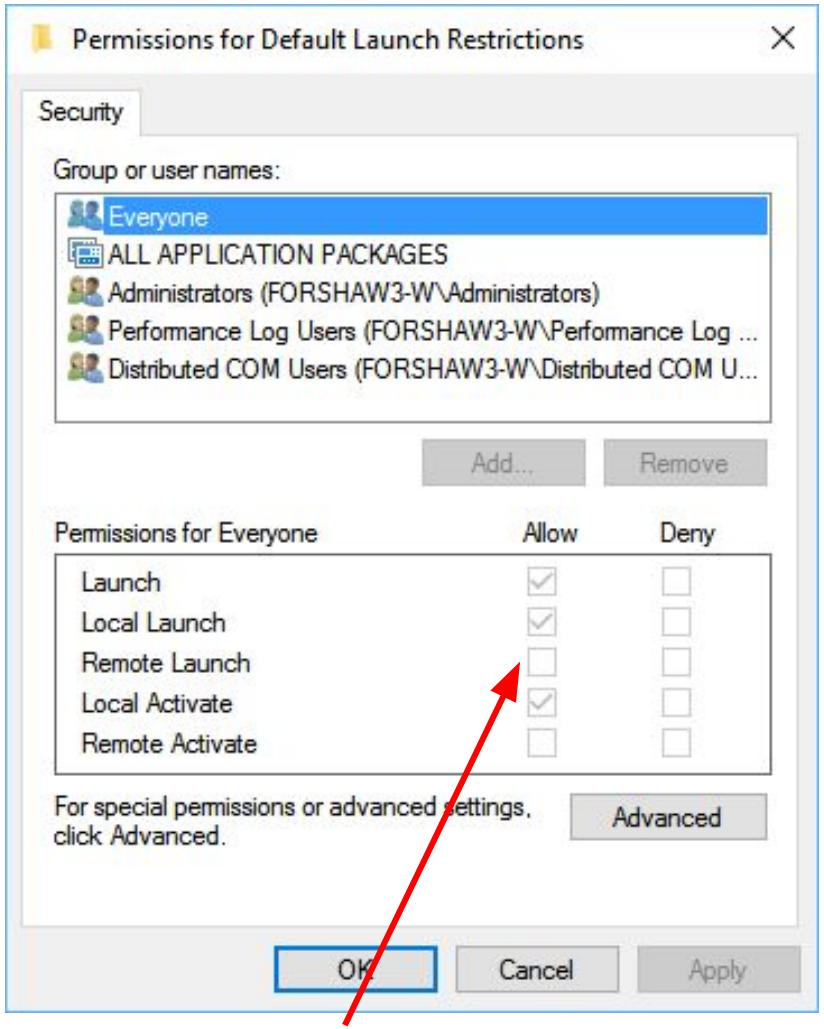
COSERVERINFO

```
struct COSERVERINFO {
  DWORD dwReserved1;
  LPWSTR pwszName;
  COAUTHINFO *pAuthInfo;
  DWORD dwReserved2;
};
```

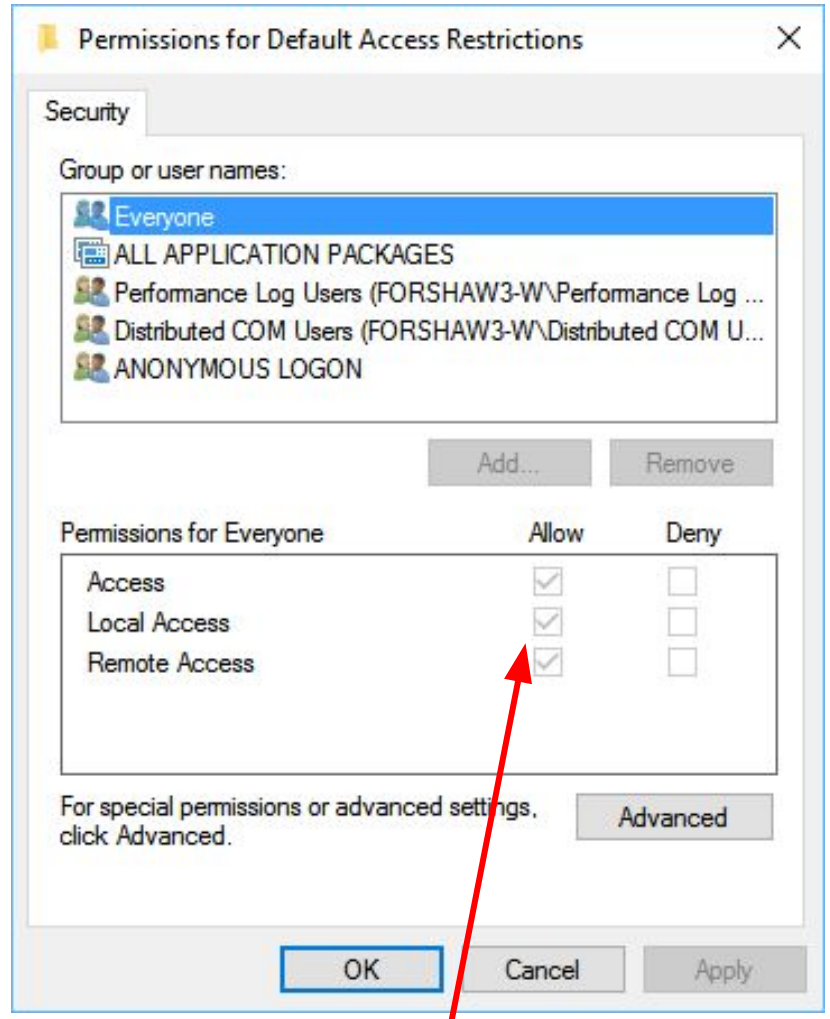
```
struct COAUTHINFO {
  DWORD dwAuthnSvc;
  DWORD dwAuthzSvc;
  LPWSTR pwszServerPrincName;
  DWORD dwAuthnLevel;
  DWORD dwImpersonationLevel;
  void* pAuthIdentityData;
  DWORD dwCapabilities;
};
```

Can specify better authentication level when connecting such as CALL or even better PKT_PRIVACY

COM Security Restrictions

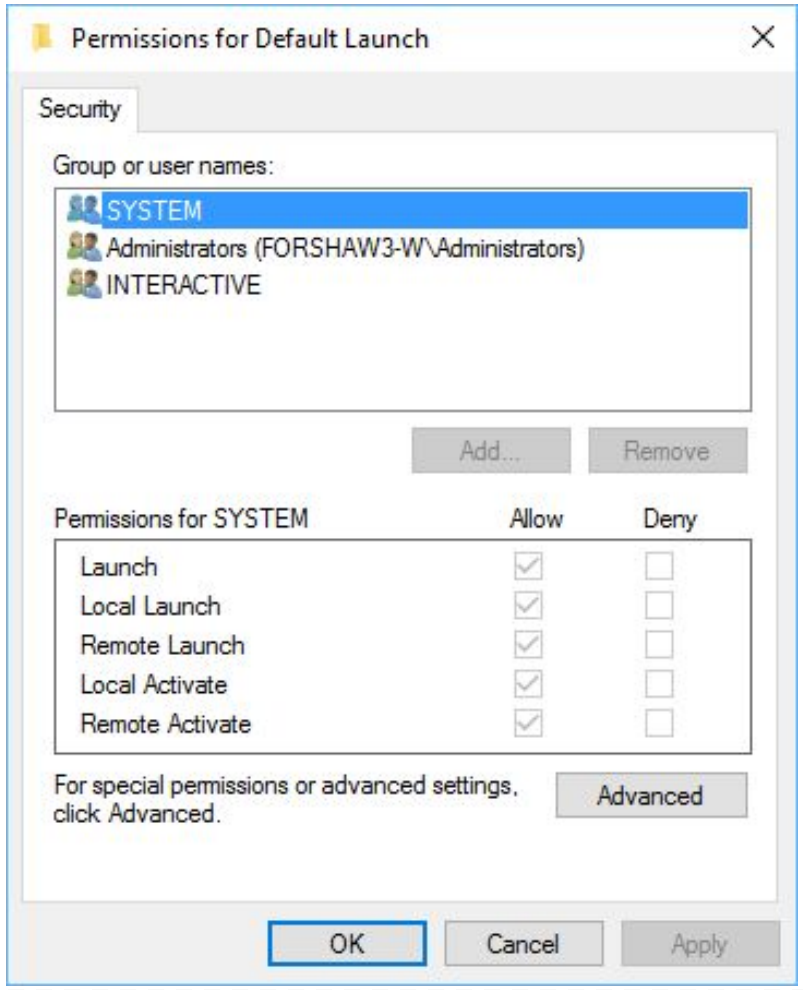


'Everyone' not allowed to launch a new object remotely

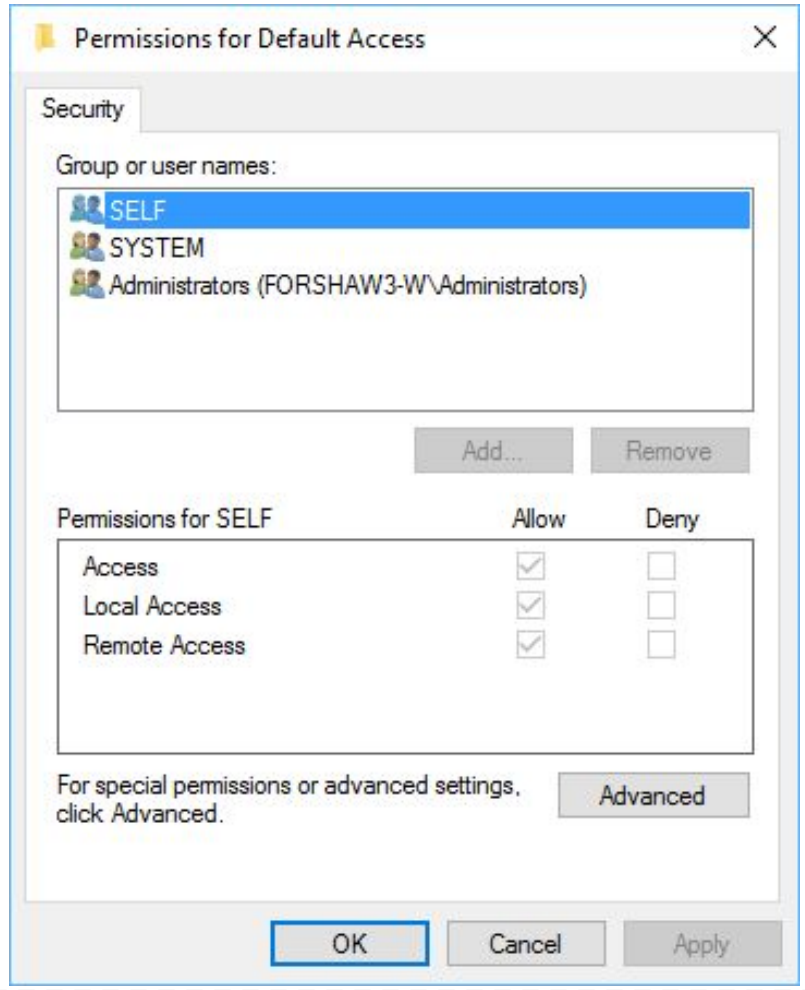


'Everyone' can access an existing object remotely

COM Security

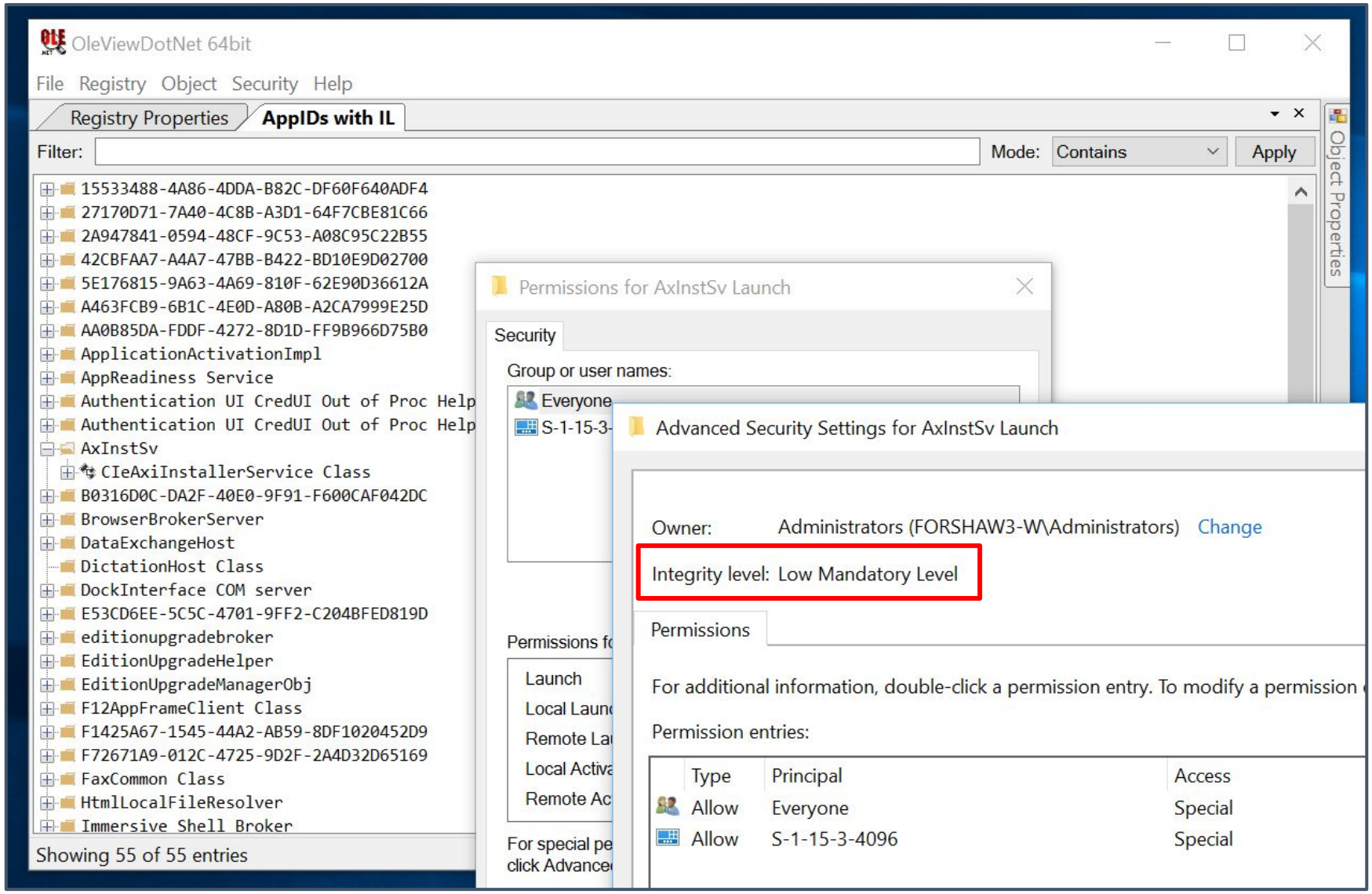


Launch = Create a new instance of the server.
Activate = Create new object on existing server.
Enforced in RPCSS



Access = Call methods on existing objects.
Enforced in Server Process
SELF = Process Token User SID

Integrity Levels



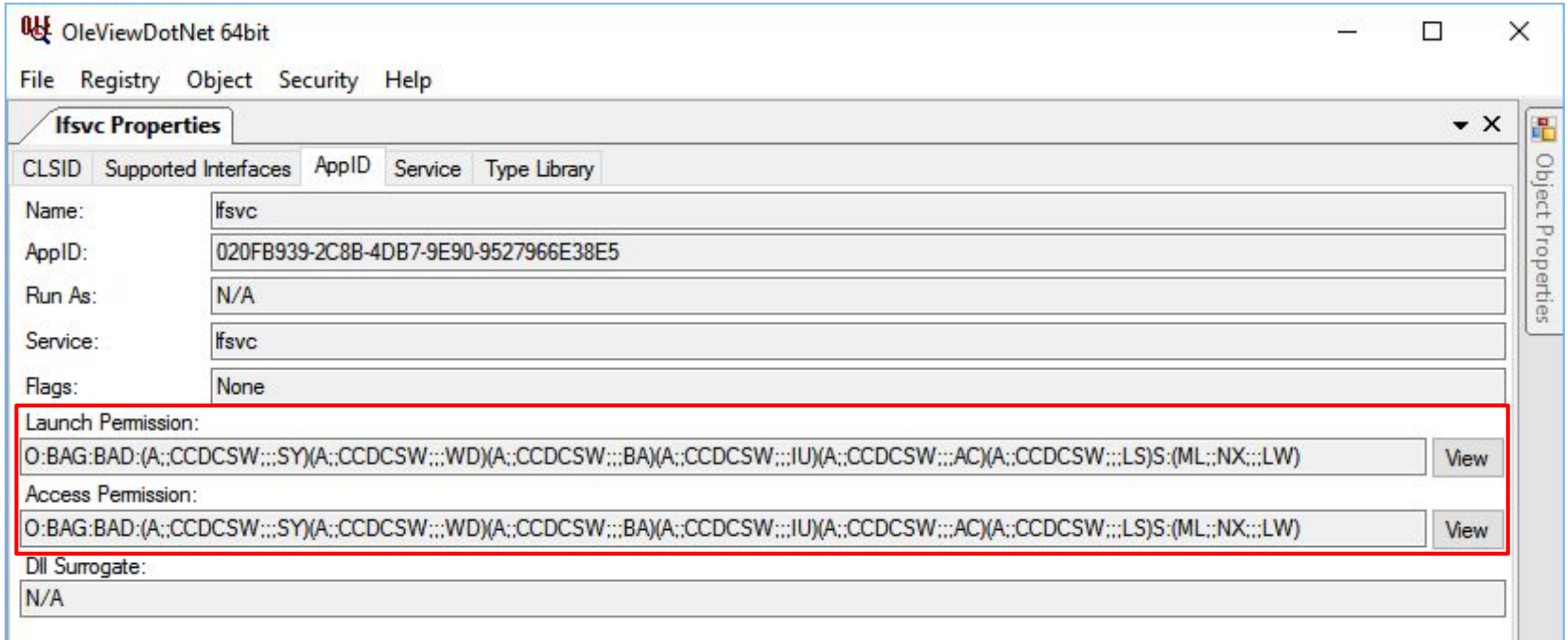
Security Through CoInitializeSecurity

```
HRESULT CoInitializeSecurity(
    PSECURITY_DESCRIPTOR pSecDesc,
    LONG cAuthSvc,
    SOLE_AUTHENTICATION_SERVICE * asAuthSvc,
    void * pReserved1,
    DWORD dwAuthnLevel,
    DWORD dwImpLevel,
    void * pAuthList,
    DWORD dwCapabilities,
    void * pReserved3
);
```

Optional SD:
NULL = No Access Security!

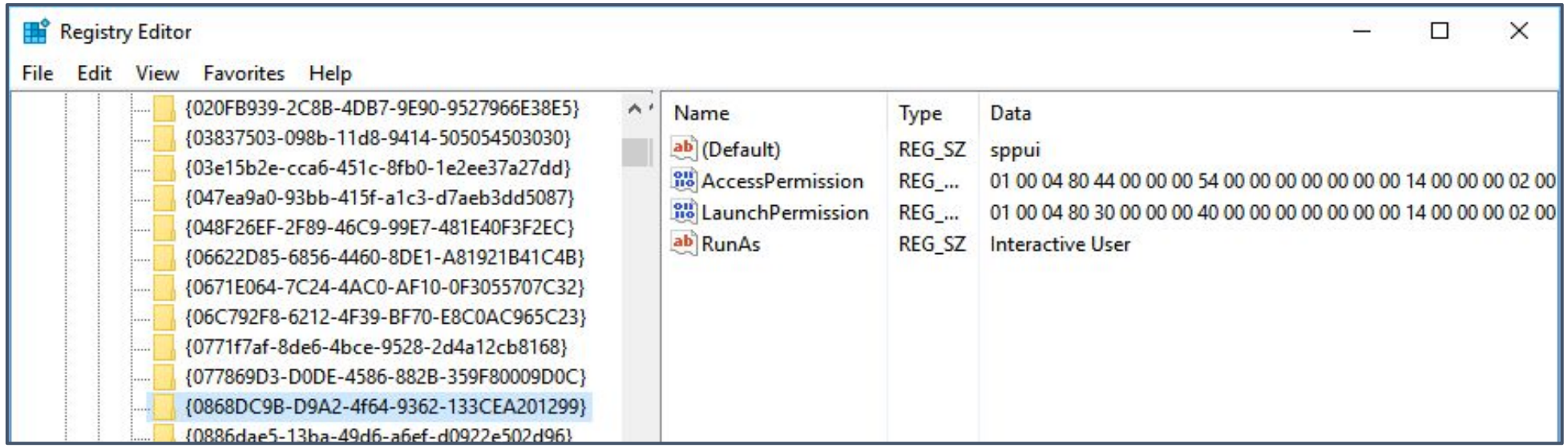
EOAC_APPID
: pSecDesc is a AppID GUID
EOAC_ACCESS_CONTROL
: pSecDesc is a pointer to an IAccessControl implementation
EOAC_NO_CUSTOM_MARSHAL
: Disables custom marshaling in the process.

Security Through AppID

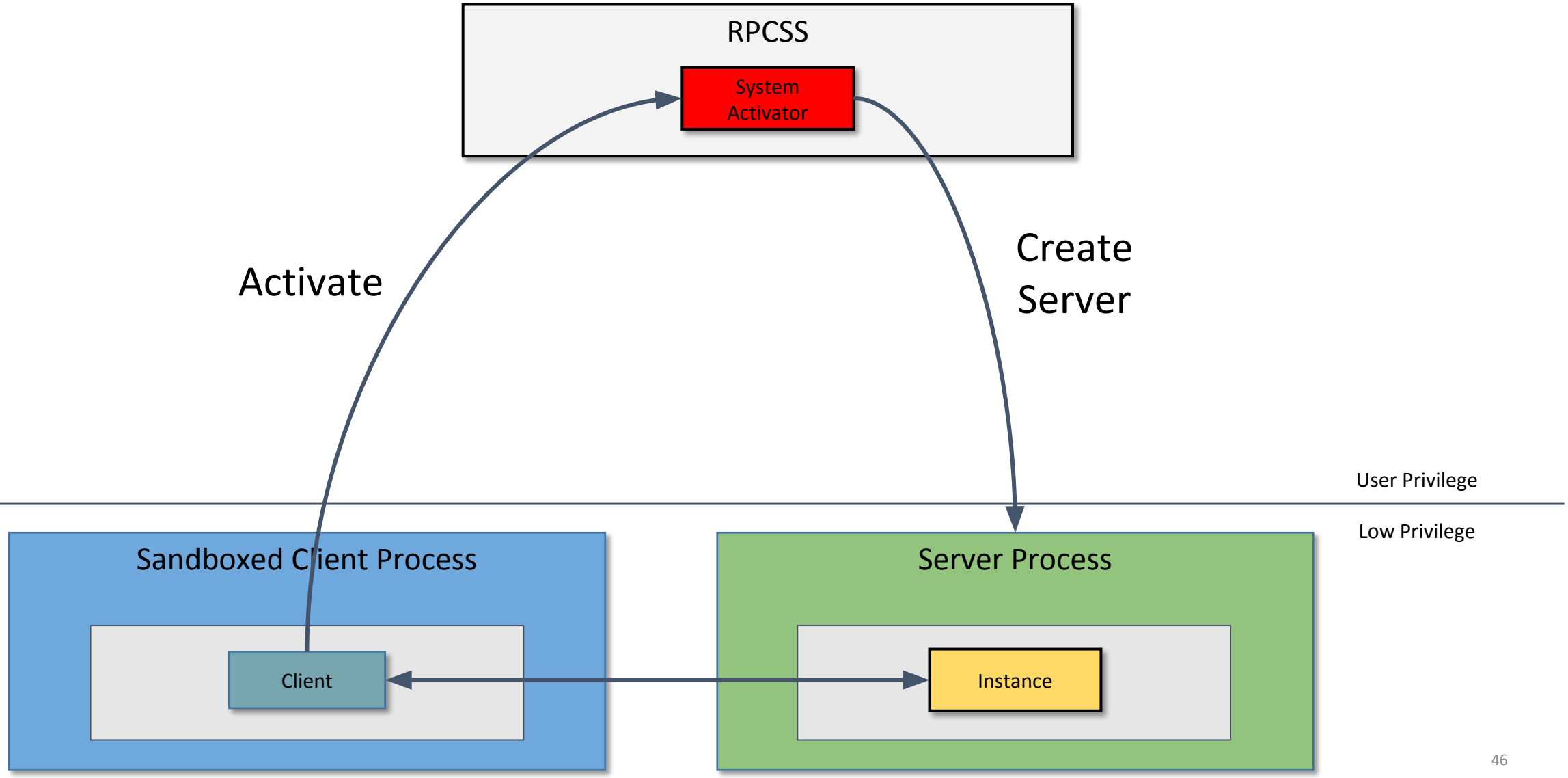


Application IDs

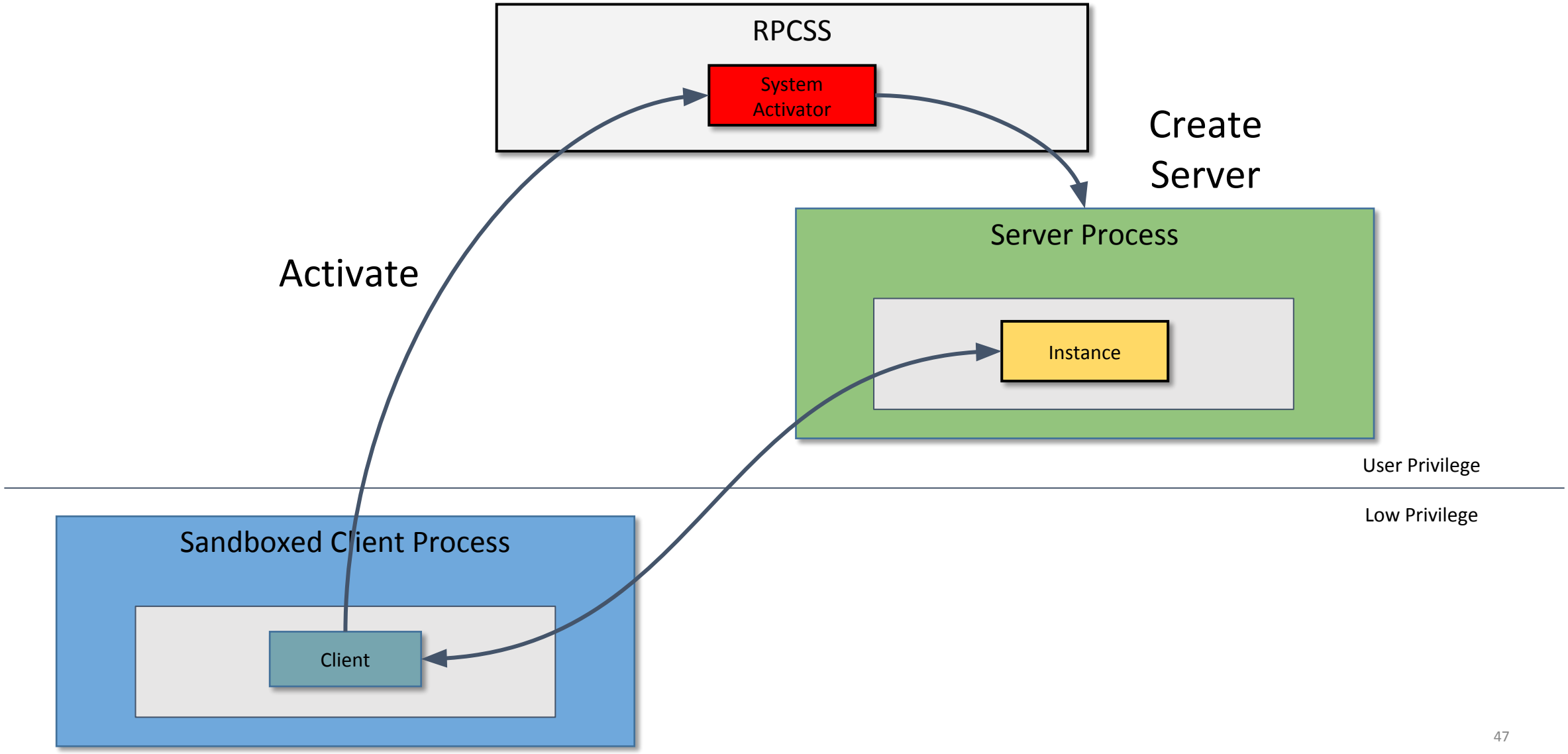
- AppIDs can configure more than just Launch/Access security
- Used to specify a number of features:
  - Allows you to specify a DLL Surrogate. This allows you to create in-process DLL servers as OOP local servers
  - Specify the object is hosted in a Windows service rather than a separate executable
  - Specify running as interactive user.



Activate as Activator (AAA)

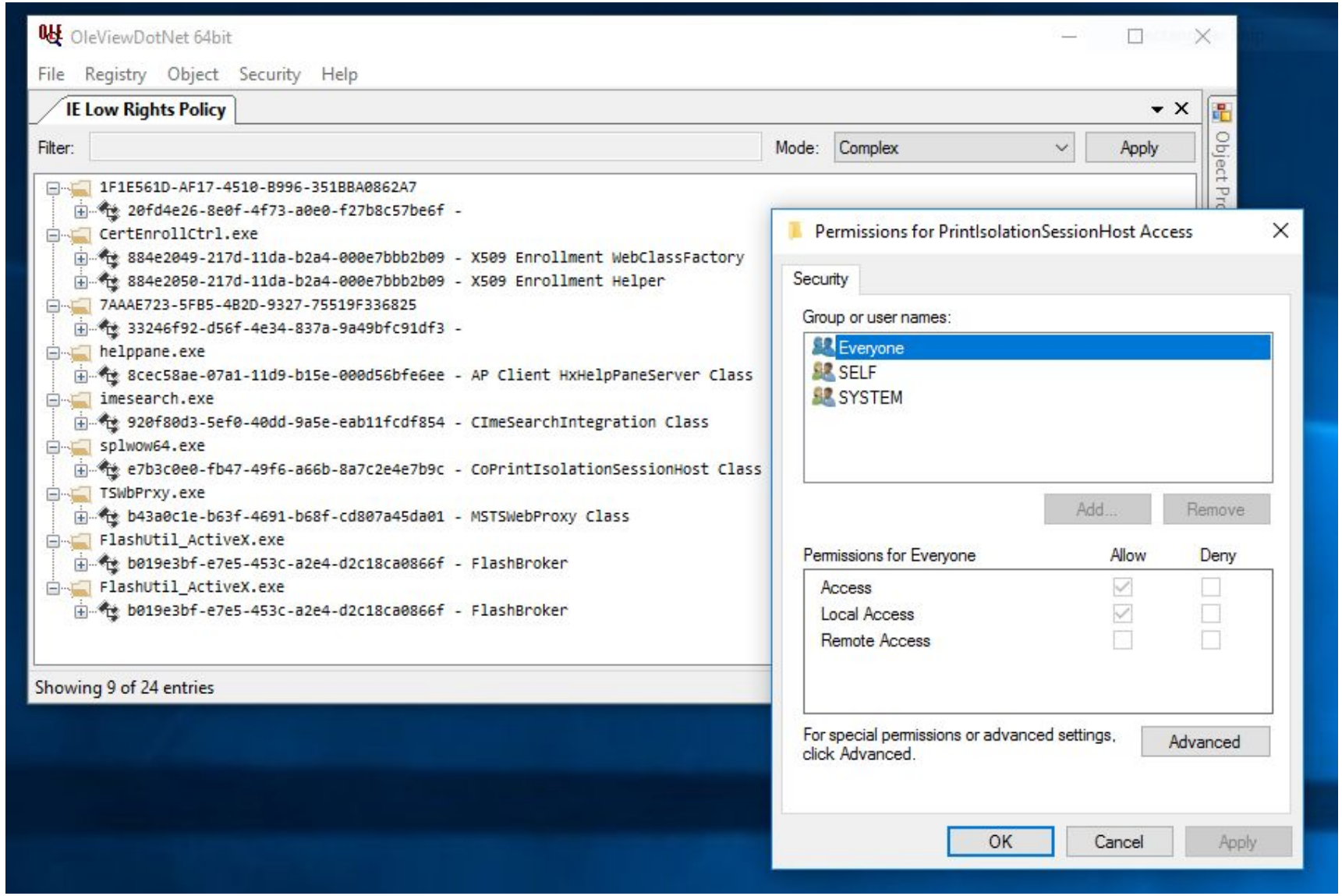


RunAs Interactive User

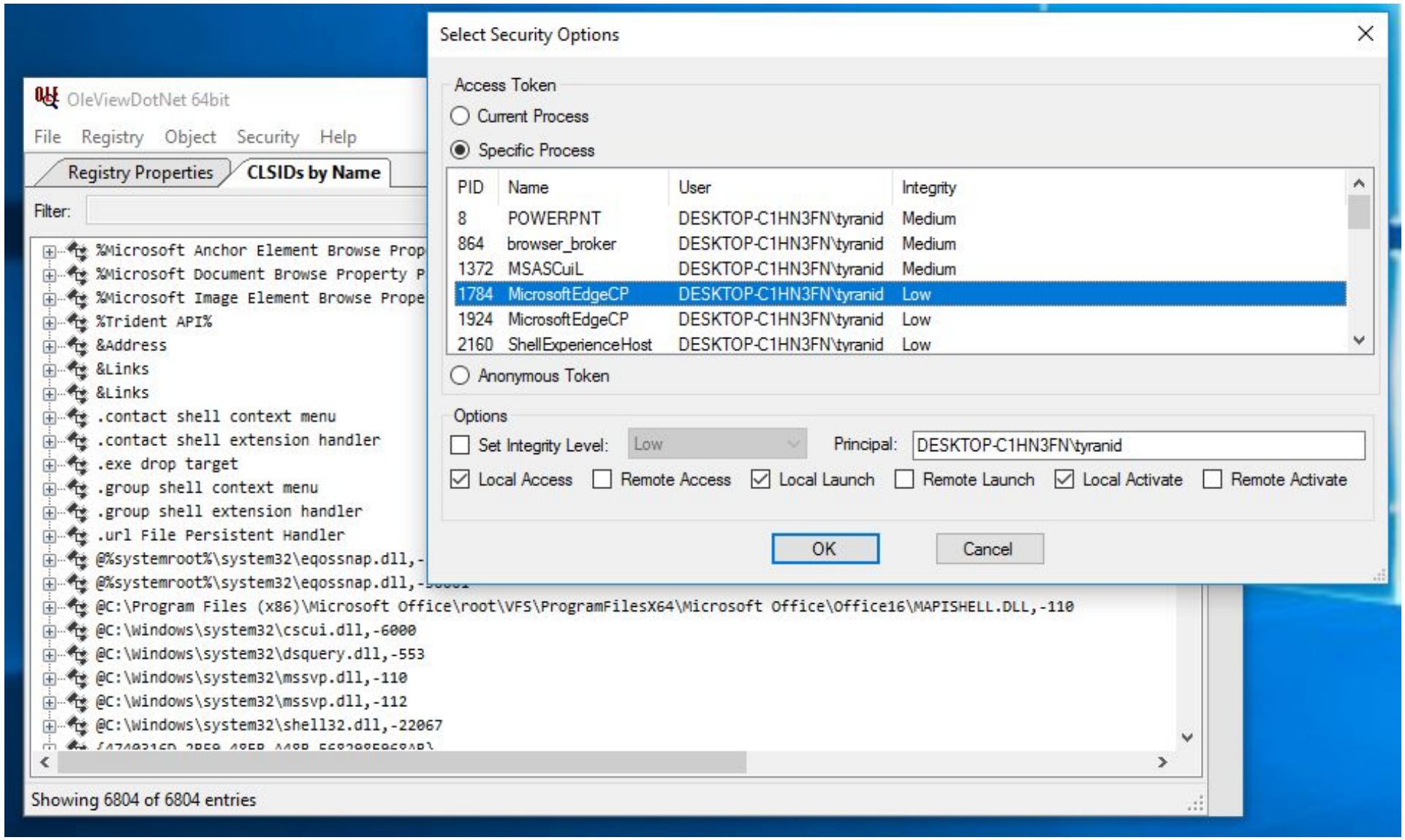


https://pixabay.com/en/lion-lion-head-teeth-background-2016620/

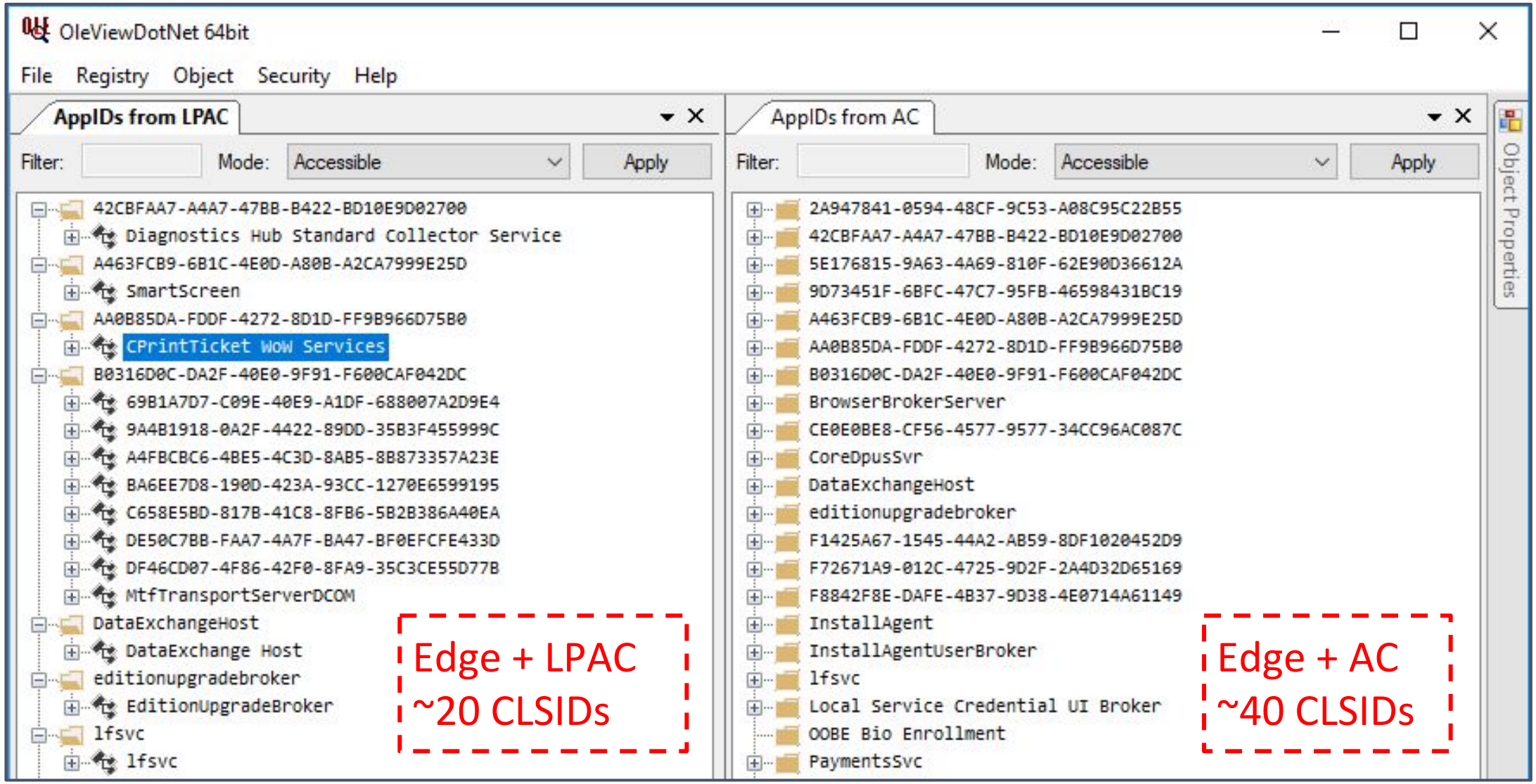
Sandbox: Activation via Broker



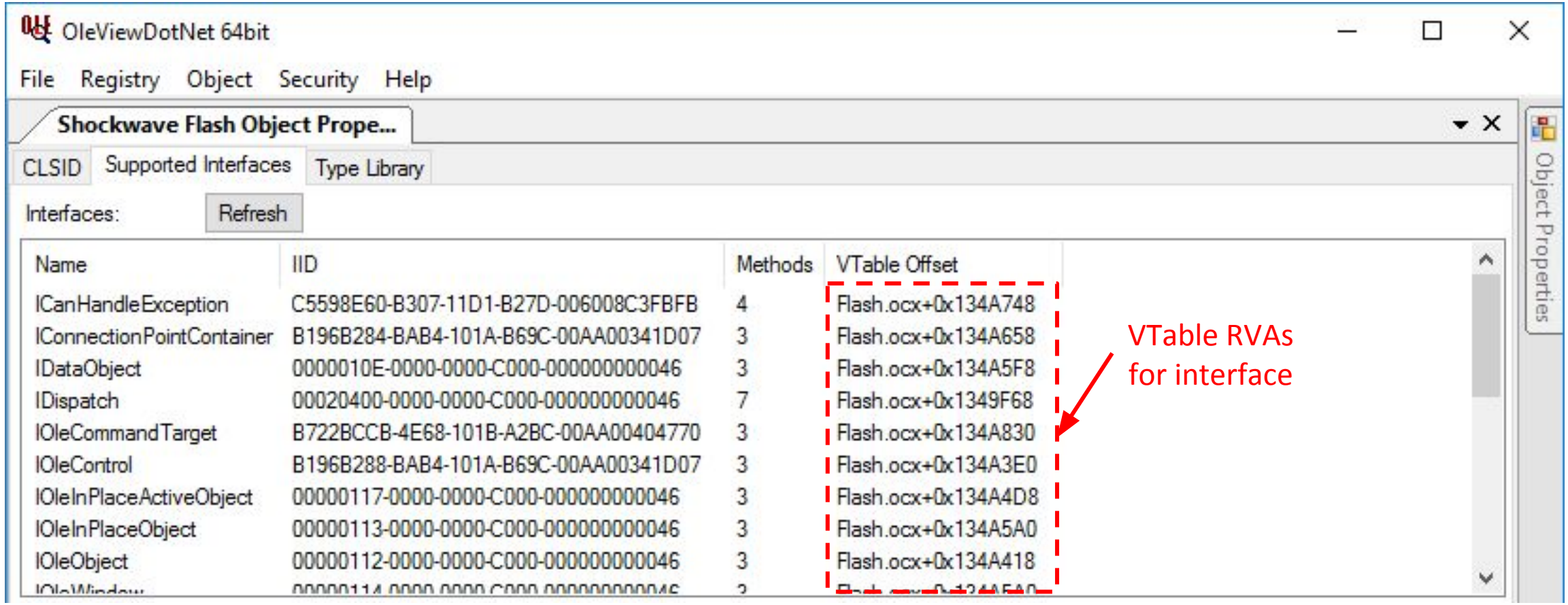
Sandbox: Activation via RPCSS



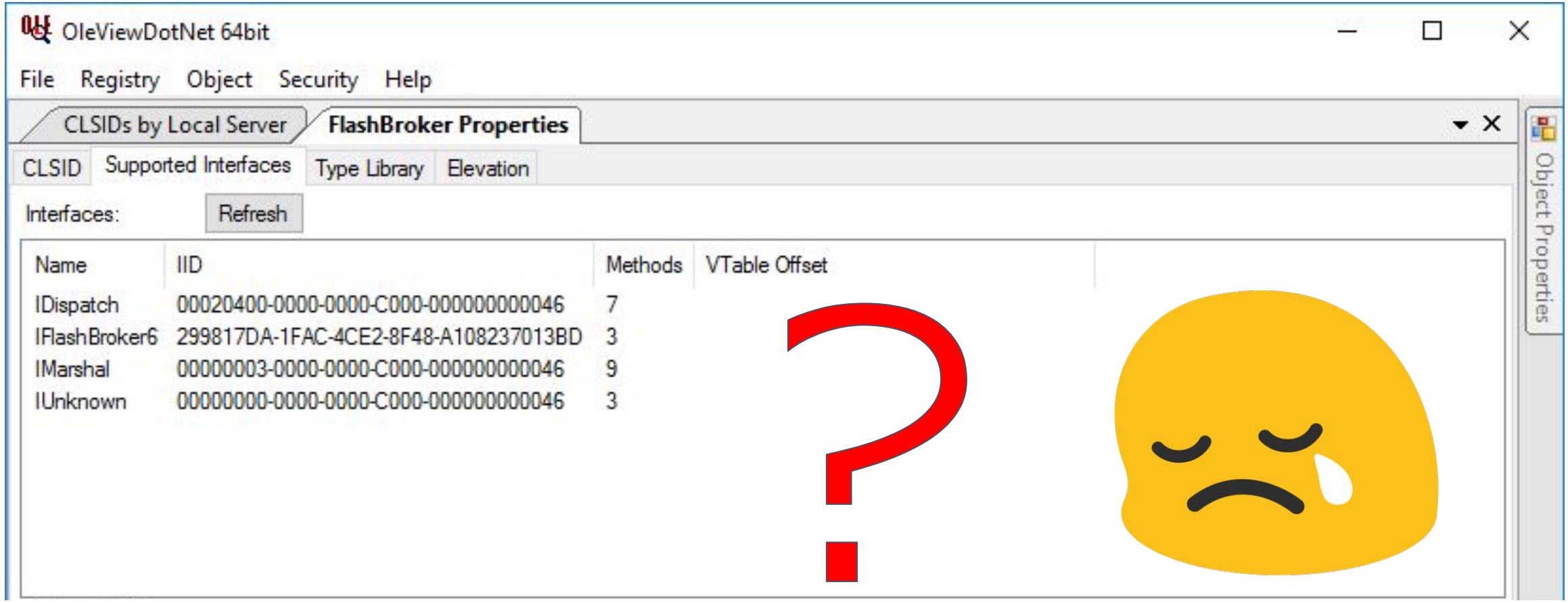
Edge is Watching You PWN



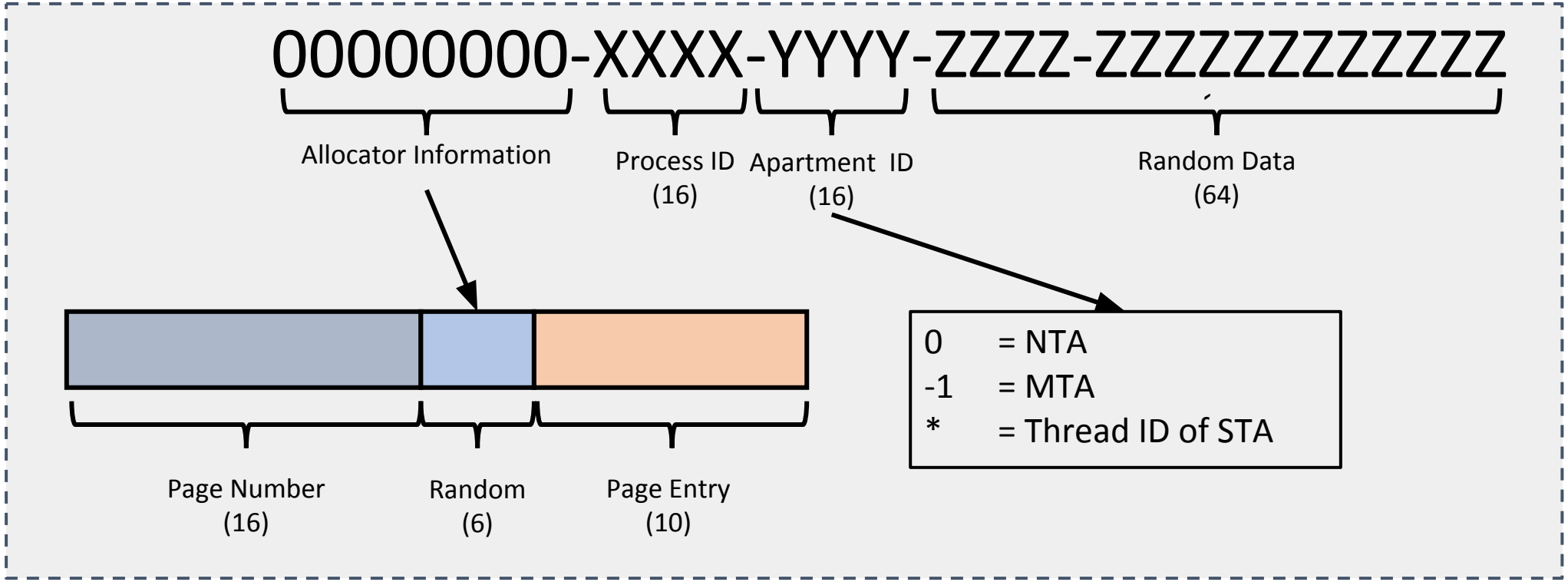
In-process Reverse Engineering



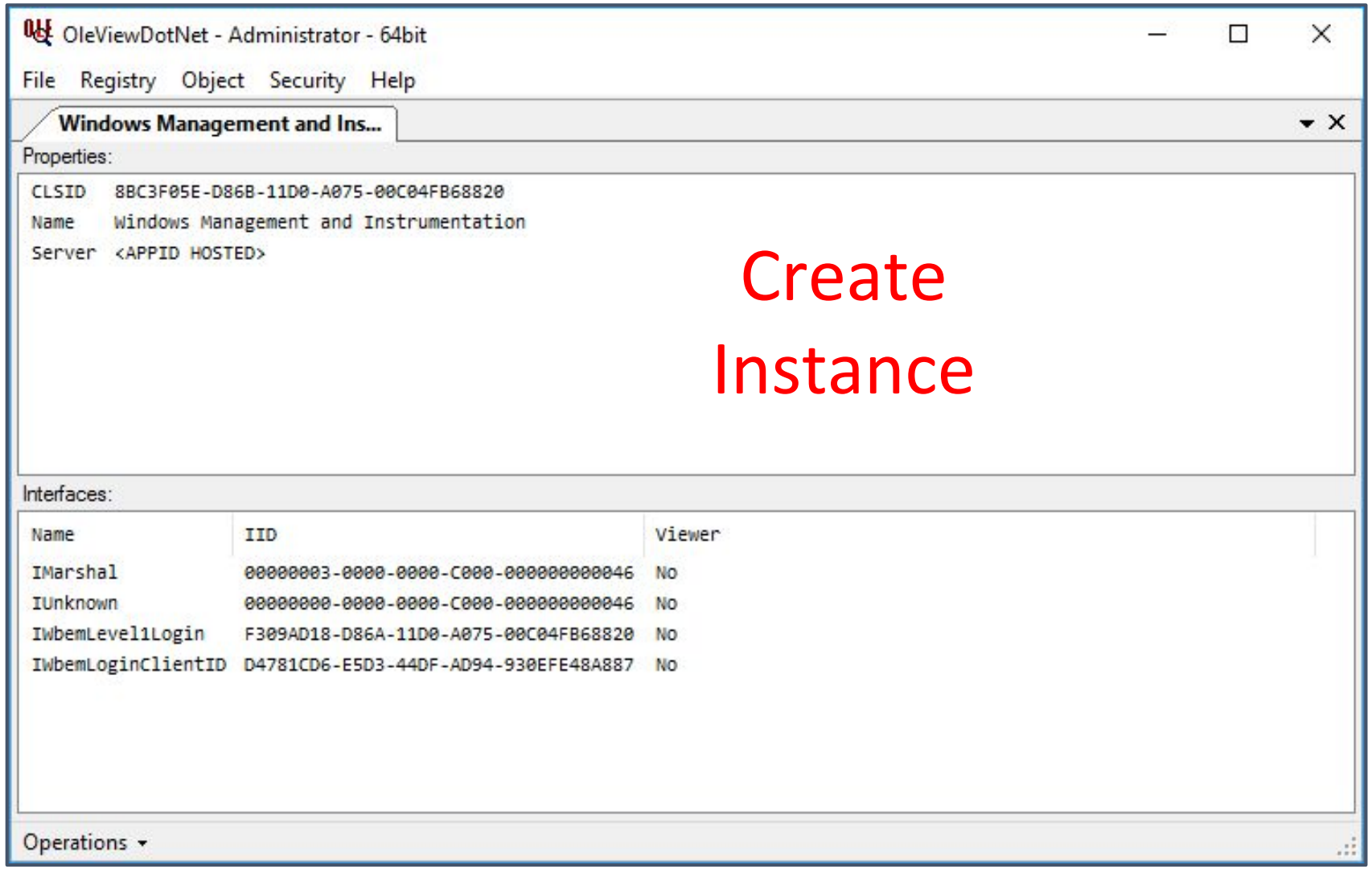
OOP Reverse Engineering



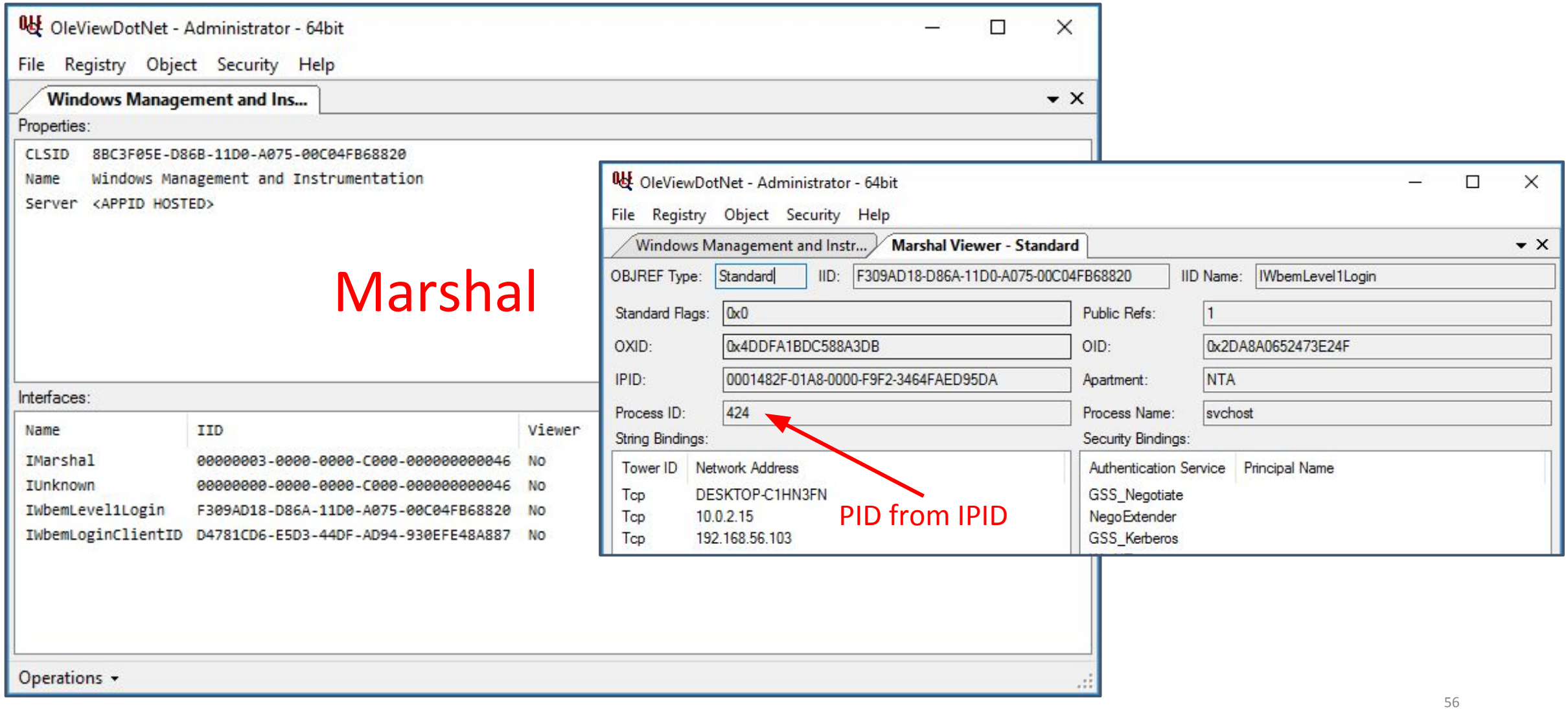
IPID Structure



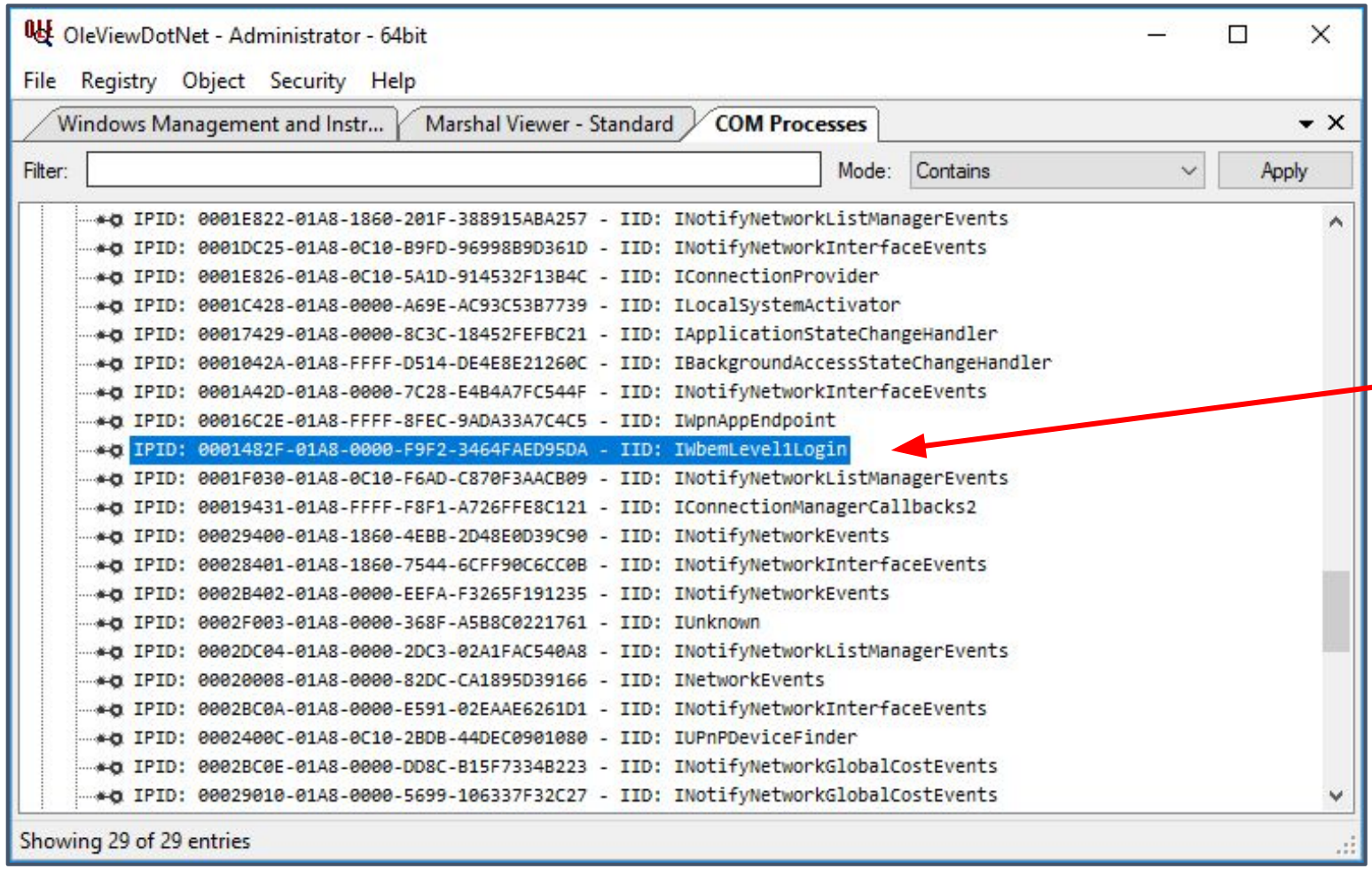
Tracking Down OOP VTable



Tracking Down OOP VTable

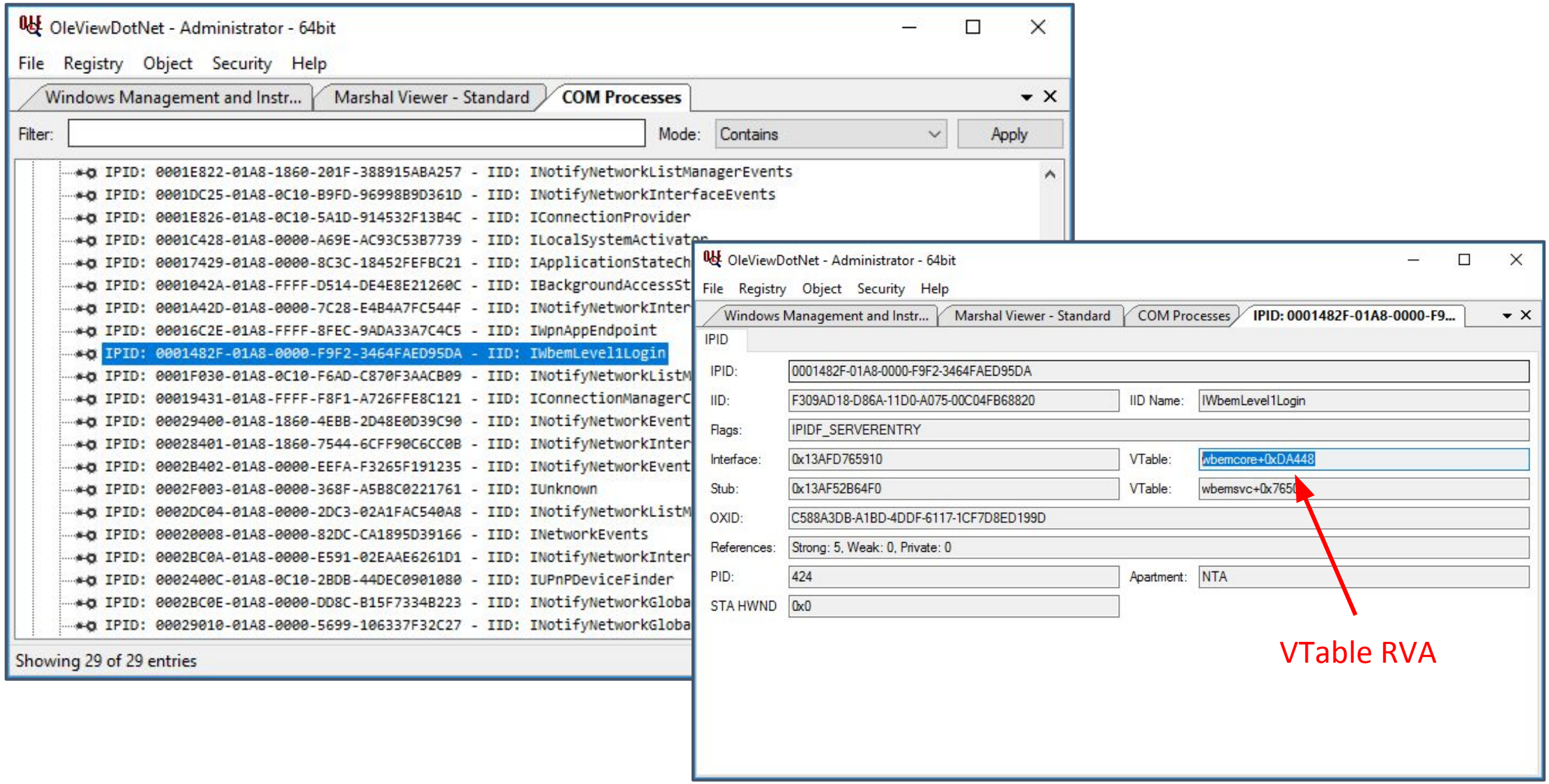


Tracking Down OOP VTable

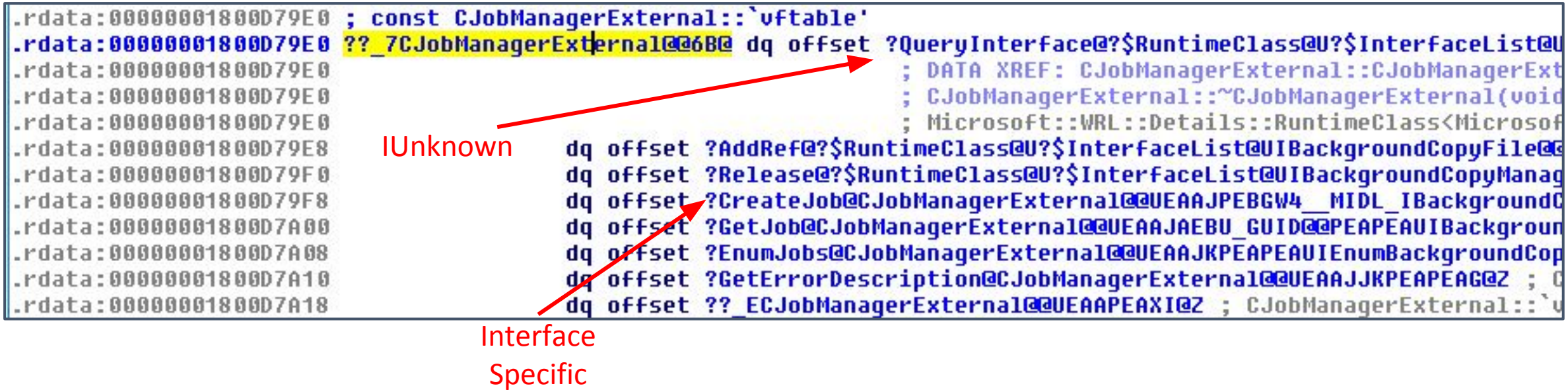


Find IPID in Process List

Tracking Down OOP VTable

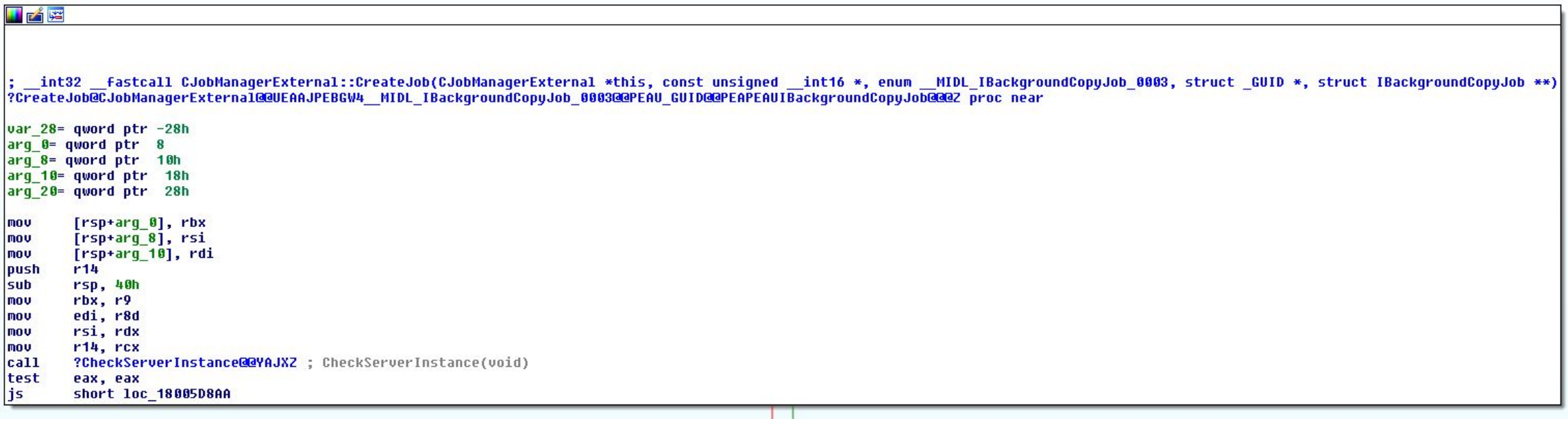


VTable Reverse Engineering



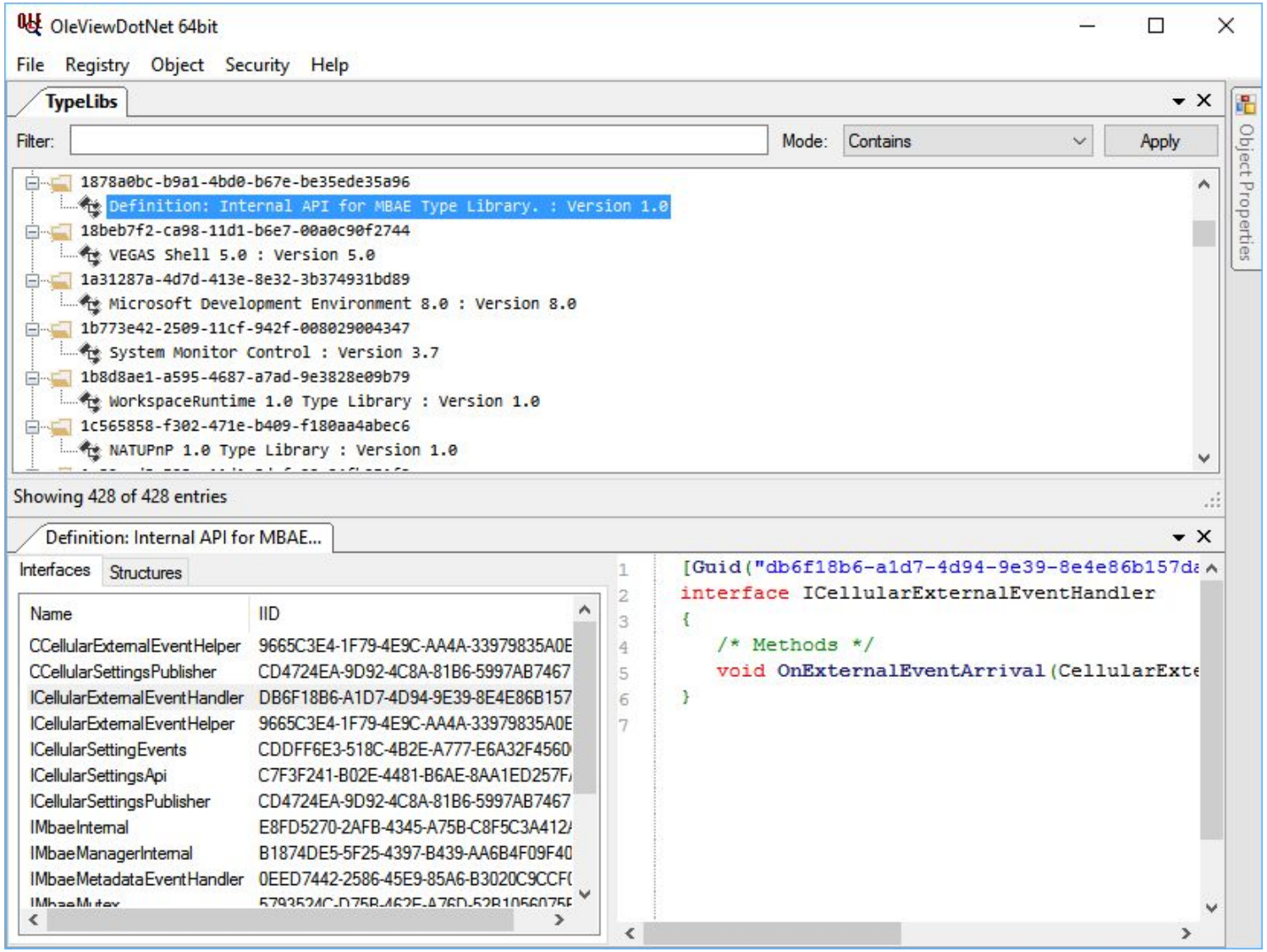
Interface Information - Public Symbols

- Most Windows components come with public symbols available
- Most COM code written in C++
- So, use C++ managed names to recover some parameter information

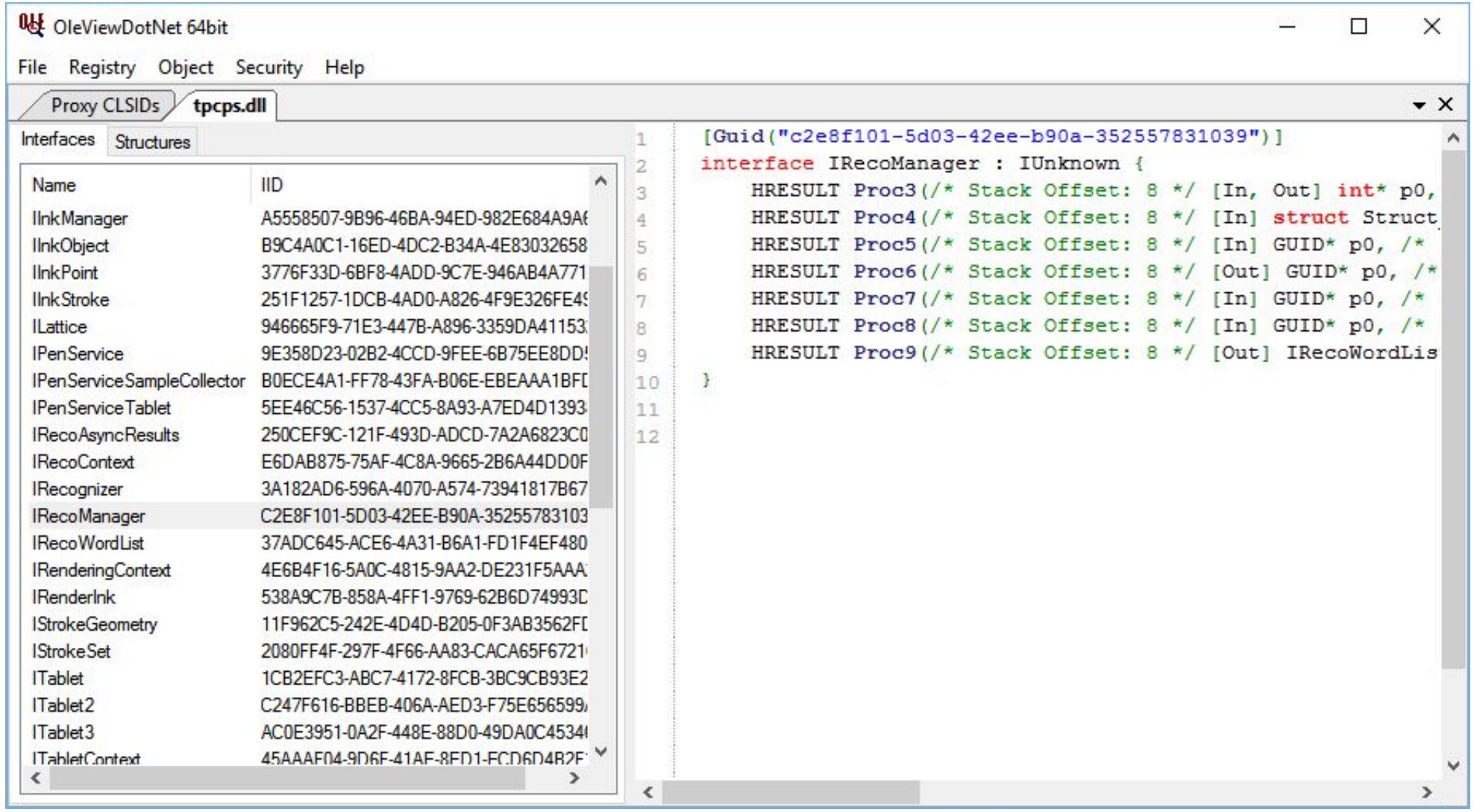


```
; __int32 __fastcall CJobManagerExternal::CreateJob(CJobManagerExternal *this, const unsigned __int16 *, enum __MIDL_IBackgroundCopyJob_0003, struct _GUID *, struct IBackgroundCopyJob **)
?CreateJob@CJobManagerExternal@@UEAAJPEBGW4__MIDL_IBackgroundCopyJob_0003@@PEAU_GUID@@PEAPEAUIBackgroundCopyJob@@@Z proc near

var_28= qword ptr -28h
arg_0= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h
arg_20= qword ptr  28h

mov     [rsp+arg_0], rbx
mov     [rsp+arg_8], rsi
mov     [rsp+arg_10], rdi
push    r14
sub     rsp, 40h
mov     rbx, r9
mov     edi, r8d
mov     rsi, rdx
mov     r14, rcx
call    ?CheckServerInstance@@YAJXZ ; CheckServerInstance(void)
test    eax, eax
js      short loc_18005D8AA
```

Interface Information - Type Libraries



Interface Information - Proxy/Stub NDR



Bugs and "Features"

https://pixabay.com/en/strip-bug-bug-macro-insect-red-812767/

Activation Properties In SPD

```
struct SpecialPropertiesData {
    unsigned long dwSessionId;
    long fRemoteThisSessionId;
    long fClientImpersonating;
    long fPartitionIDPresent;
    DWORD dwDefaultAuthnLvl;
    GUID guidPartition;
    DWORD dwPRTFlags;
    DWORD dwOrigClsctx;
    DWORD dwFlags;
    DWORD dwPid;
    unsigned __int64 hwnd;
    DWORD ulServiceId;
    DWORD Reserved[4];
};
```

Choosing a Session ID?

UAC Related Stuff

```
enum SPD_FLAGS {
  SPD_FLAG_USE_CONSOLE_SESSION,
  SPD_FLAG_USE_DEFAULT_AUTHN_LVL,
  SPD_FLAG_USE_SERVER_PID,
  SPD_FLAG_USE_LUA_LEVEL_ADMIN,
  SPD_FLAG_COAUTH_USER_IS_NULL,
  SPD_FLAG_COAUTH_DOMAIN_IS_NULL,
  SPD_FLAG_COAUTH_PWD_IS_NULL,
  SPD_FLAG_USE_LUA_LEVEL_HIGHEST
};
```

Session and Elevation Monikers

Session-to-Session Activation with a Session Moniker

Session-to-session activation (also called cross-session activation) allows a client process to start (activate) a local server process on a specified session. This feature is available for applications that are configured to run in the security context of the interactive user, also known as the "RunAs Interactive User" object activation mode. For more information about security contexts, see The Client's Security Context.

Distributed COM (DCOM) enables object activation on a per-session basis by using a system-supplied Session Moniker. Other system-supplied monikers include file monikers, item monikers, generic composite monikers, anti-monikers, pointer monikers, and URL monikers.

The COM Elevation Moniker

The COM elevation moniker allows applications that are running under user account control (UAC) to activate COM classes with elevated privileges. For more information, see **Focus on Least Privilege**.

When to Use the Elevation Moniker

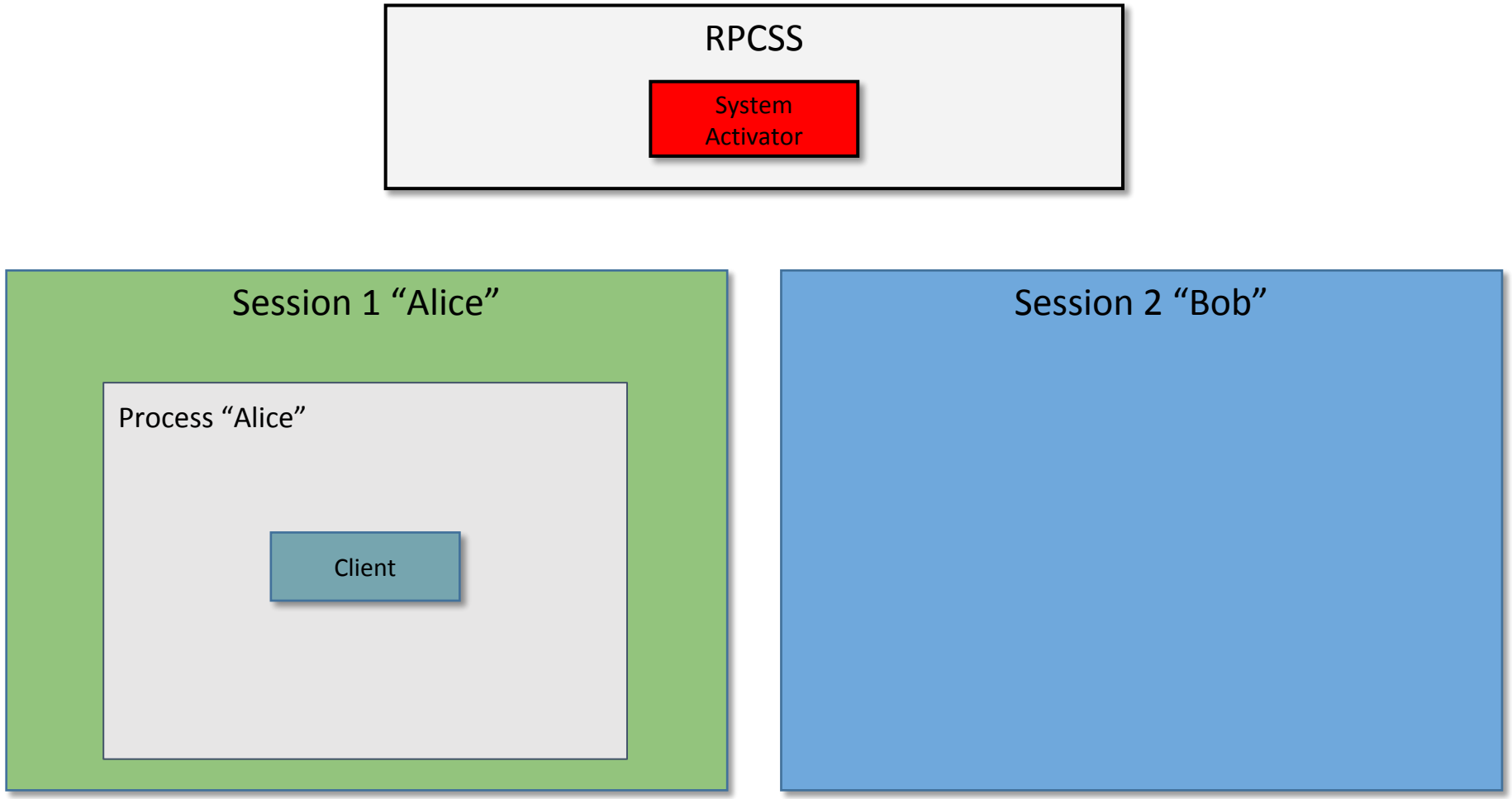
The elevation moniker is used to activate a COM class to accomplish a specific and limited function that requires elevated privileges, such as changing the system date and time.

Elevation requires participation from both a COM class and its client. The COM class must be configured to support elevation by annotating its registry entry, as described in the Requirements section. The COM client must request elevation by using the elevation moniker.

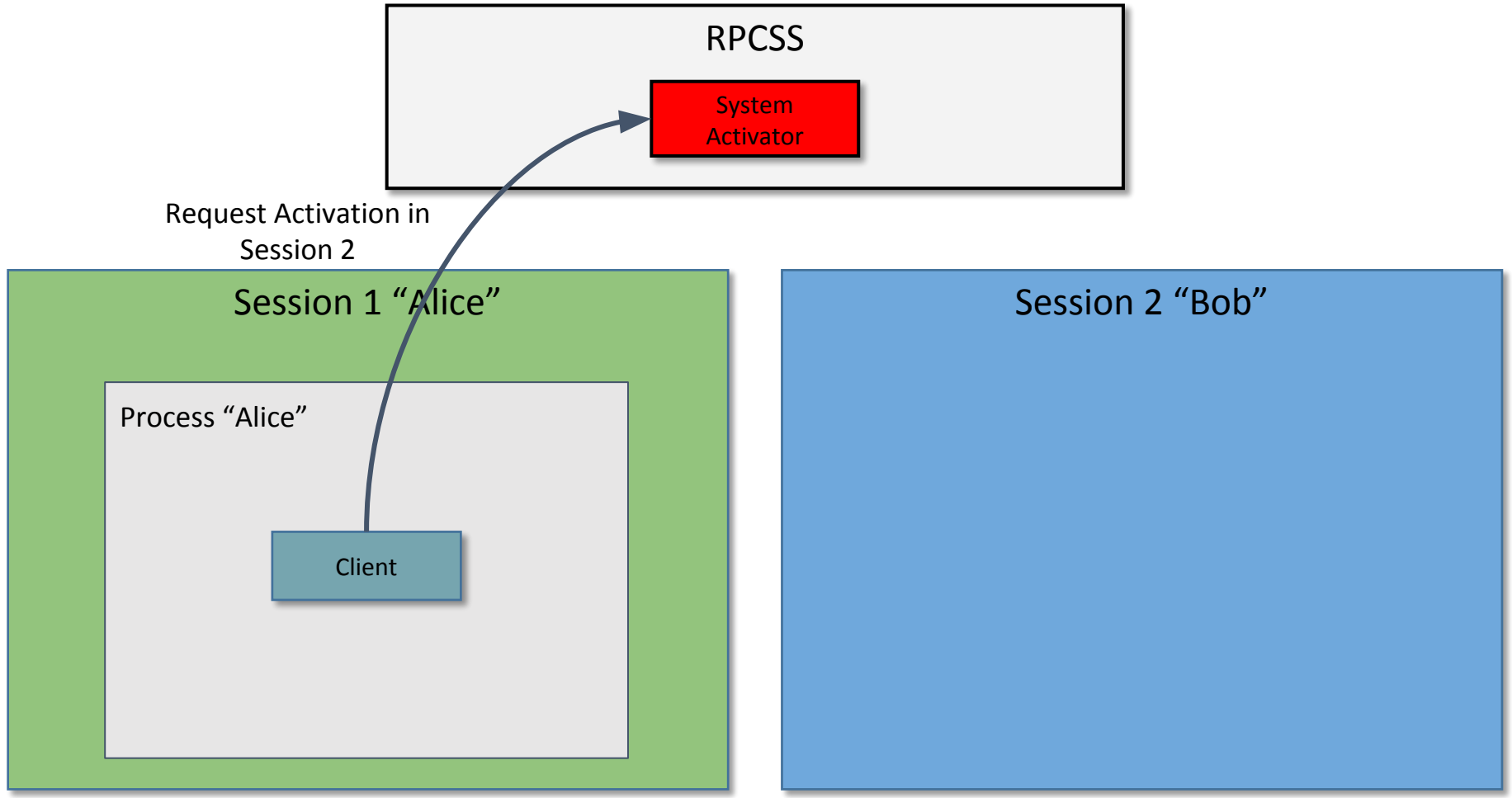
Monikers and Binding

```
HRESULT CoCreateInstanceInSession(DWORD session,
    REFCLSID rclsid, REFIID riid, void ** ppv) {
    BIND_OPTS3 bo = {};
    WCHAR  wszCLSID[50];
    WCHAR  wszMonikerName[300];

    StringFromGUID2(rclsid, wszCLSID, _countof(wszCLSID));
    StringCchPrintf(wszMonikerName, _countof(wszMonikerName),
      L"session:%d!new:%s", session, wszCLSID);
    bo.cbStruct = sizeof(bo);
    bo.dwClassContext  = CLSCTX_LOCAL_SERVER;
    return CoGetObject(wszMonikerName, &bo, riid, ppv);
}
```

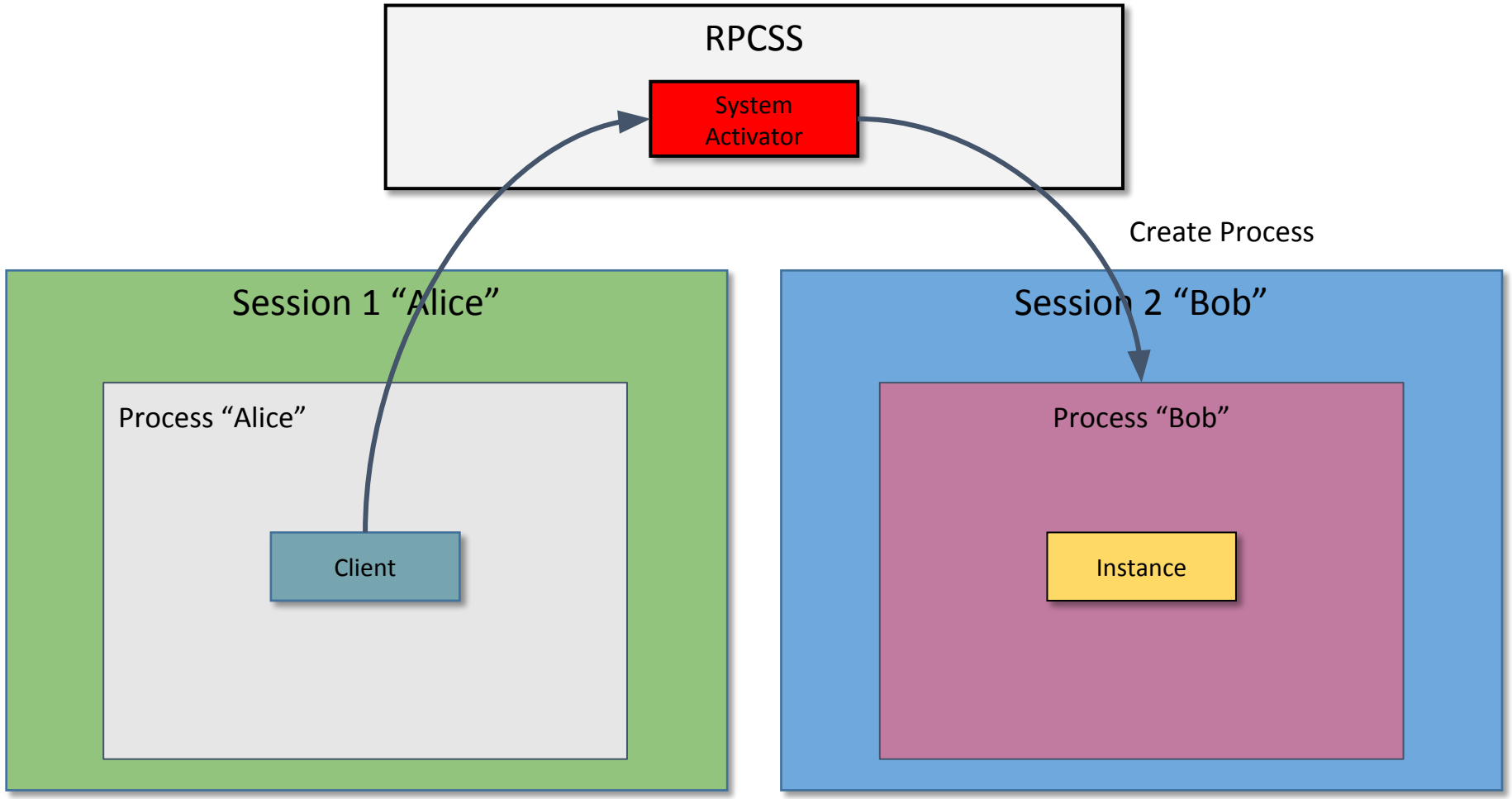
Session Moniker in Action



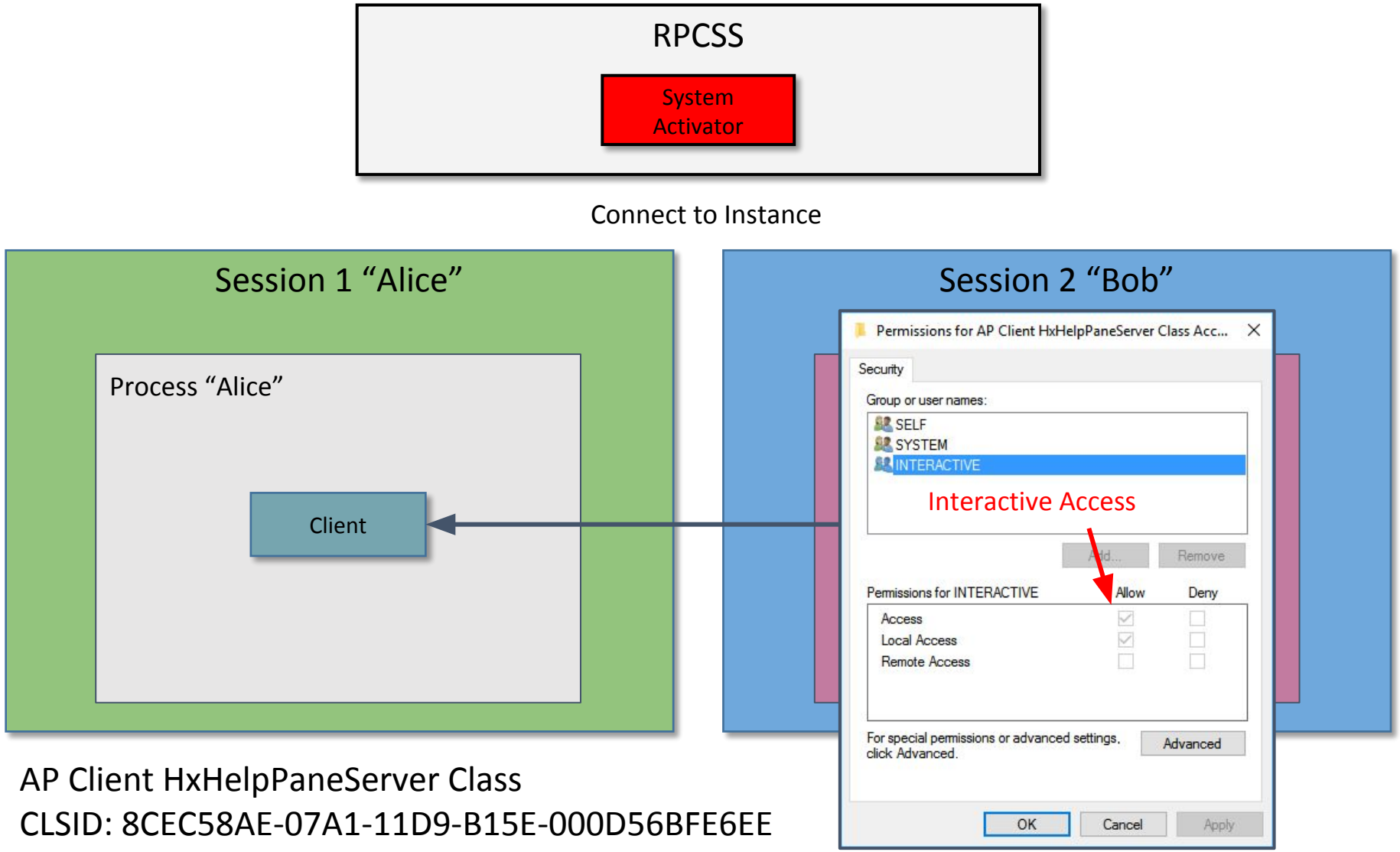
Session Moniker in Action



Session Moniker in Action

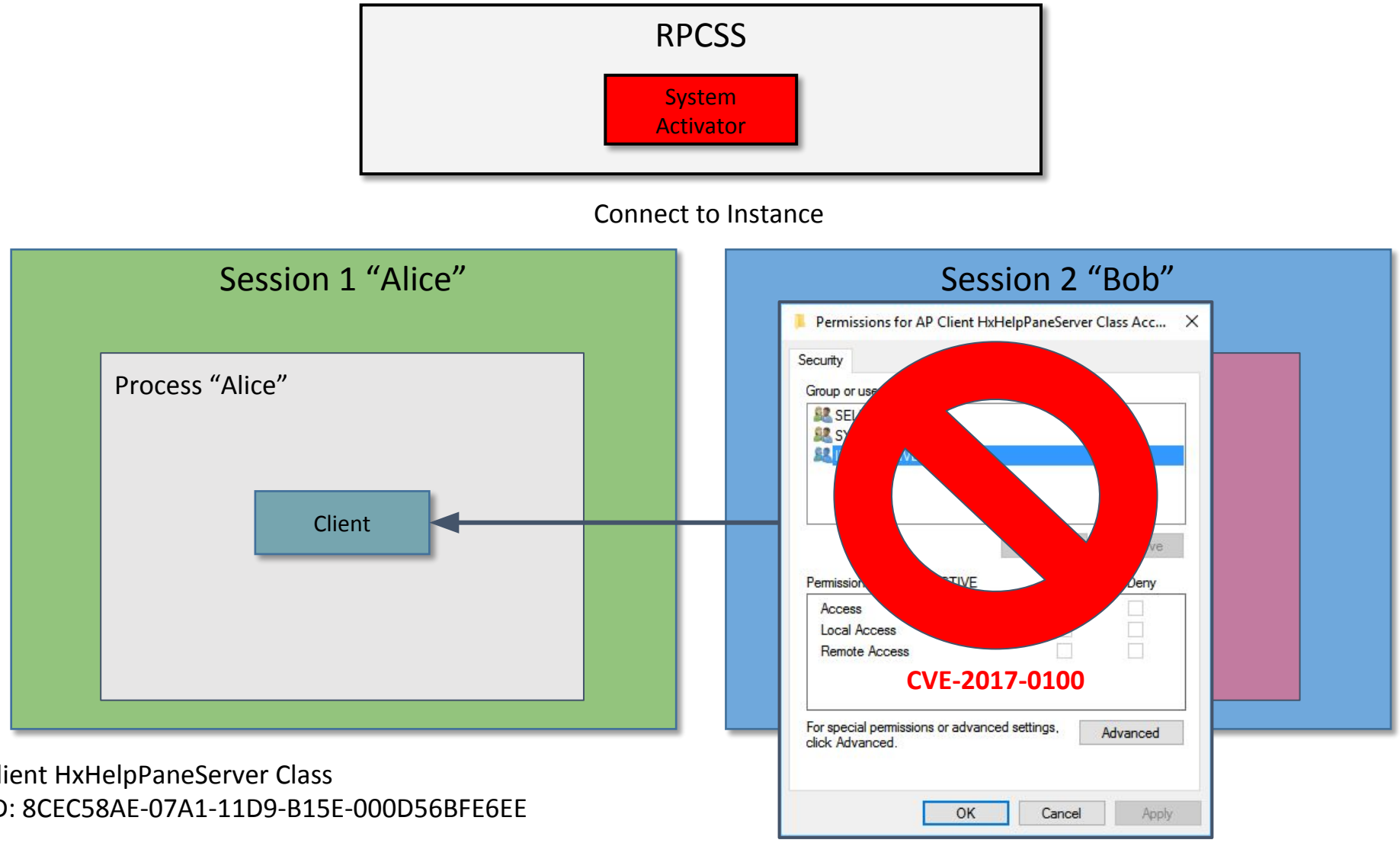


Session Moniker in Action



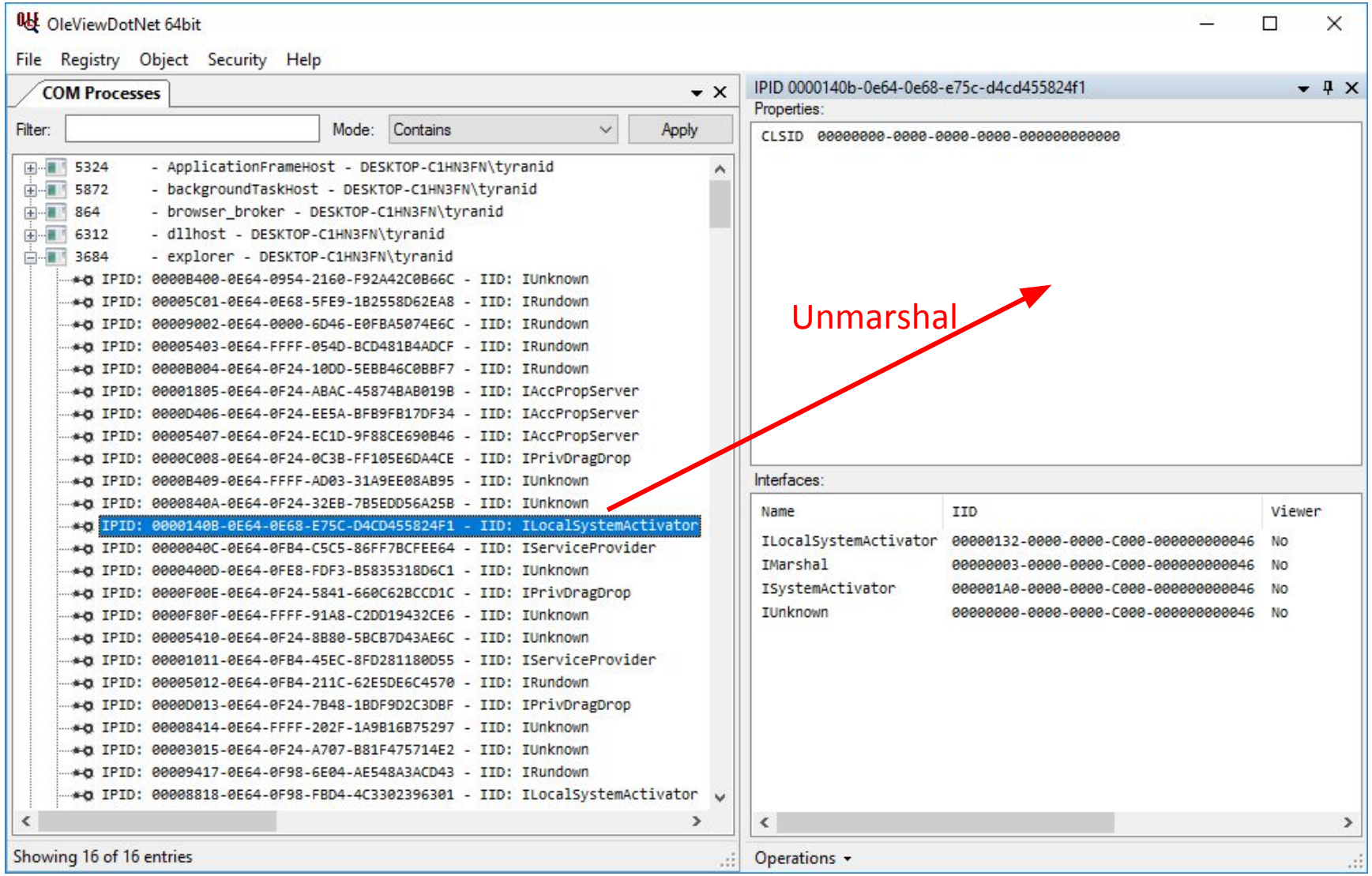
AP Client HxHelpPaneServer Class
CLSID: 8CEC58AE-07A1-11D9-B15E-000D56BFE6EE

Session Moniker in Action



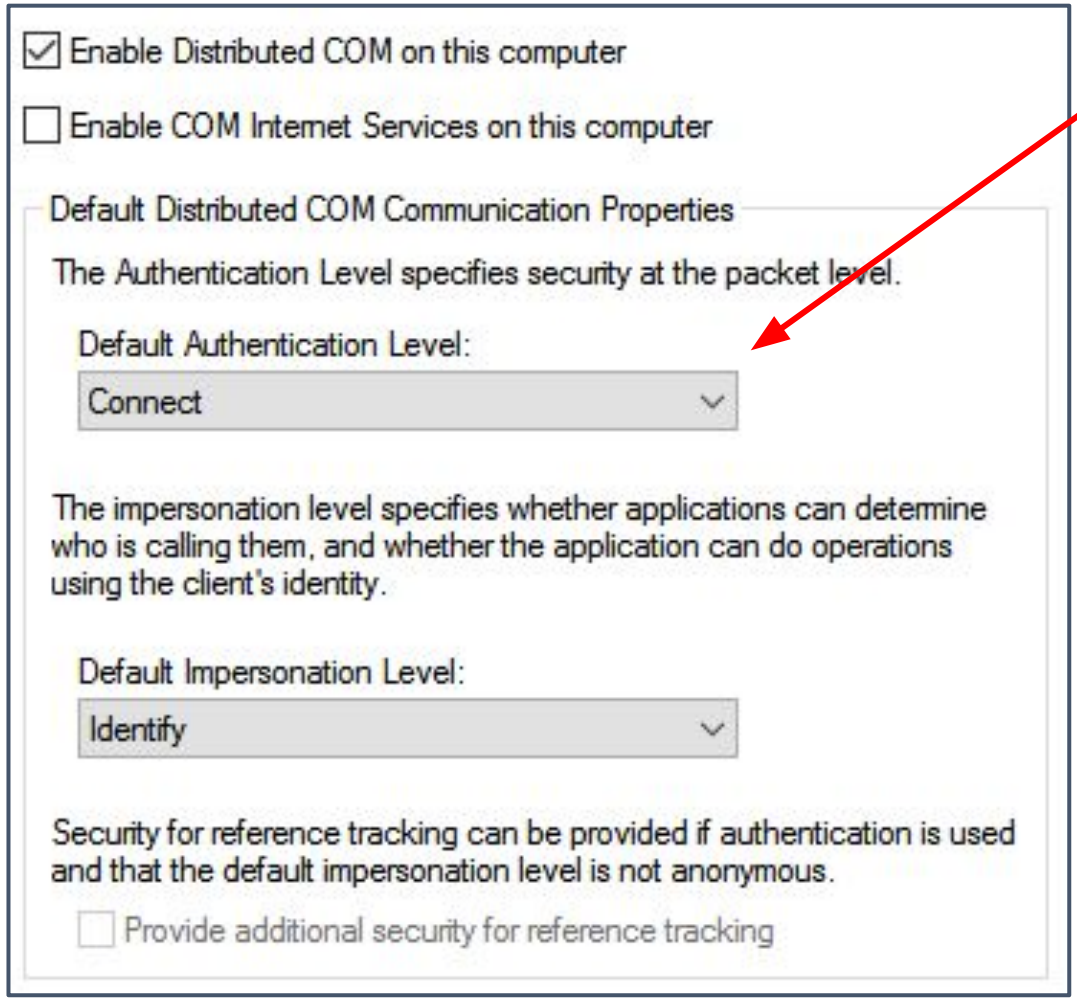
AP Client HxHelpPaneServer Class
CLSID: 8CEC58AE-07A1-11D9-B15E-000D56BFE6EE

Abusing Information Disclosure



You Can’t Help Loving DCOM

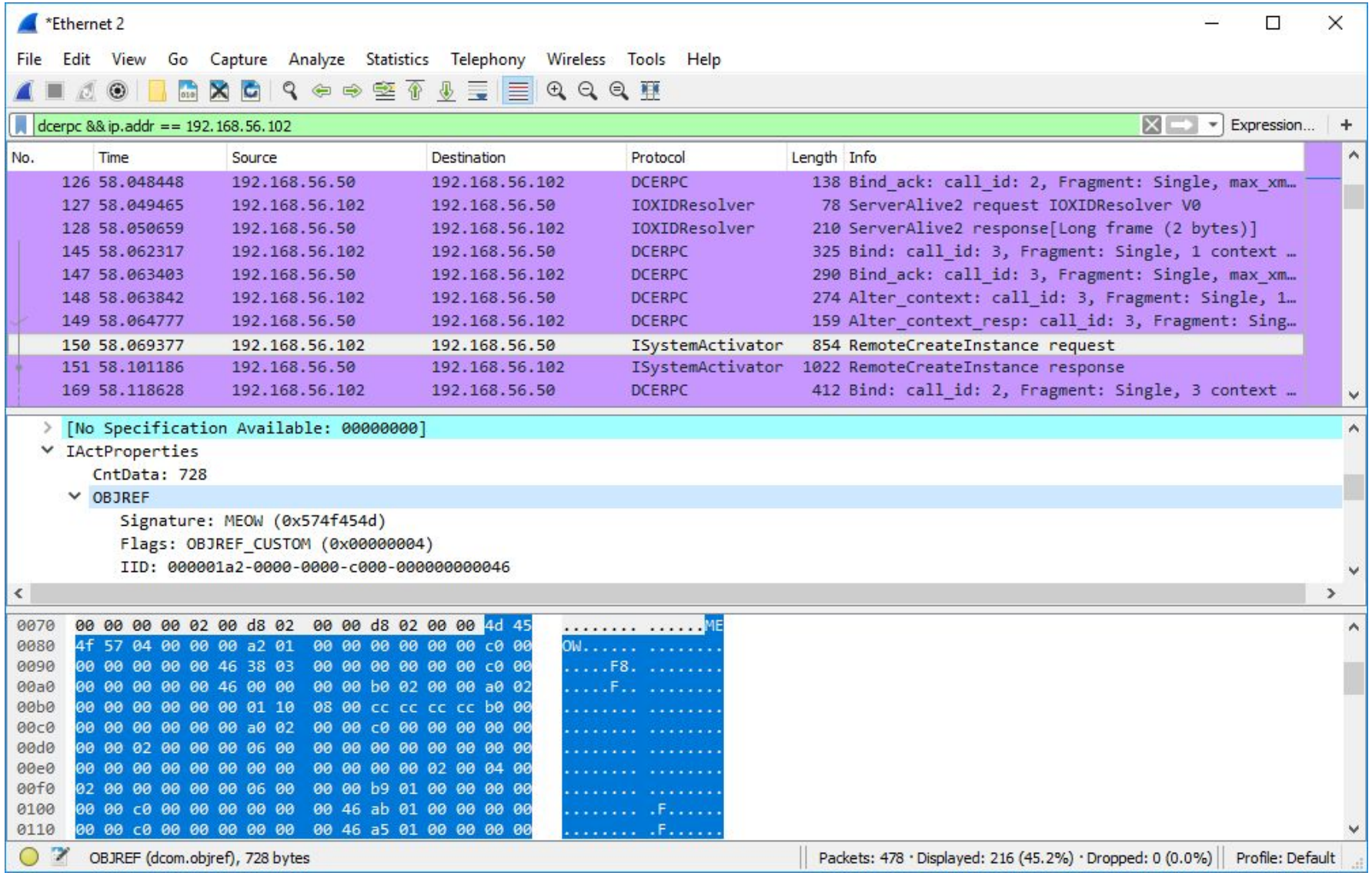
Plain Text Communications?



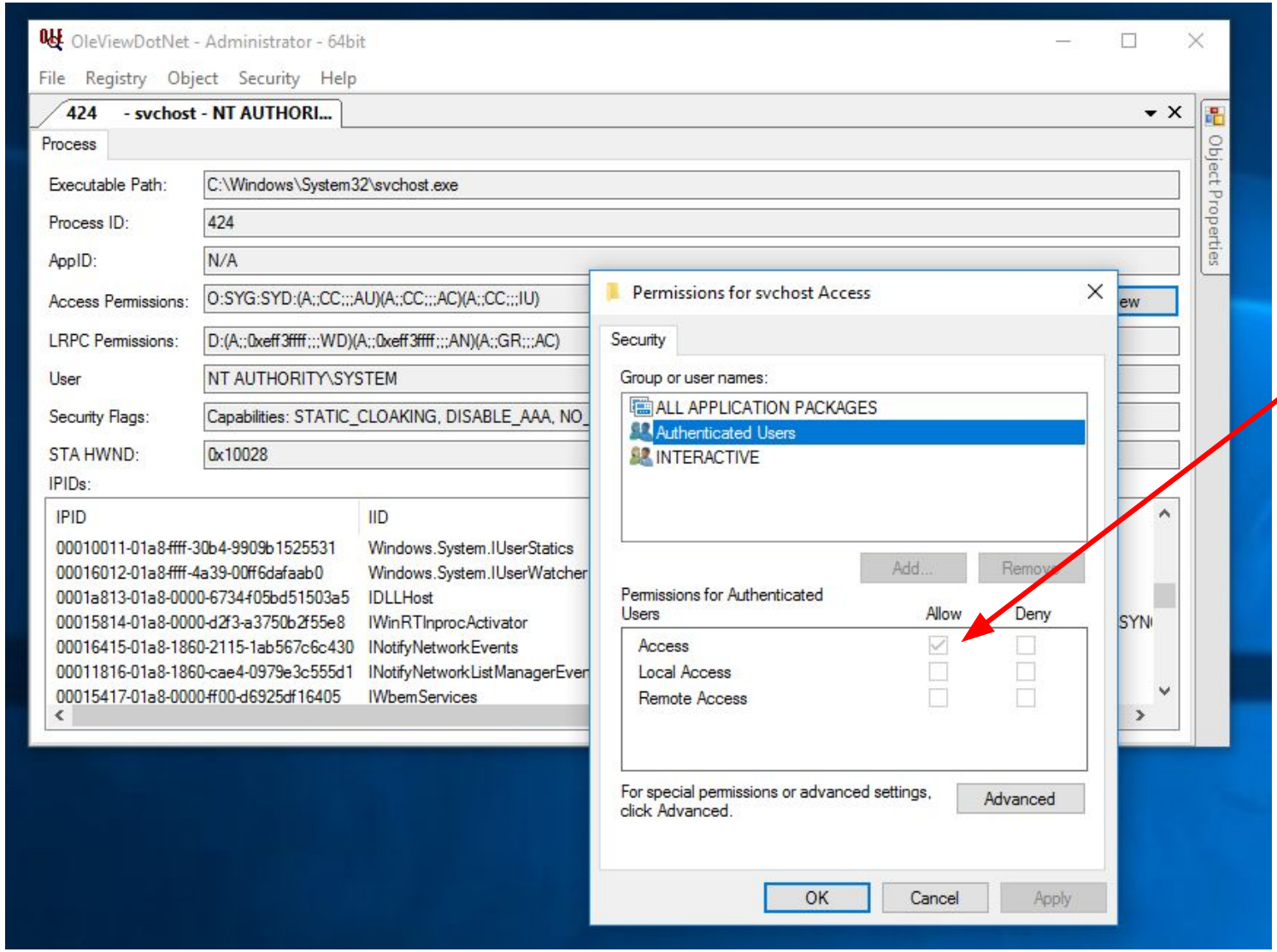
Defaults to CONNECT Authentication Level?

Authentication Level	
CONNECT	Authenticates the credentials of the client and server.
CALL/PKT	Same as CONNECT but also prevents replay attacks.
PKT_INTEGRITY	Same as CALL/PKT but also verifies that none of the data transferred between the client and server has been modified.
PKT_PRIVACY	Same as PKT_INTEGRITY but also ensures that the data transferred can only be seen unencrypted by the client and the server.

Good old Wireshark

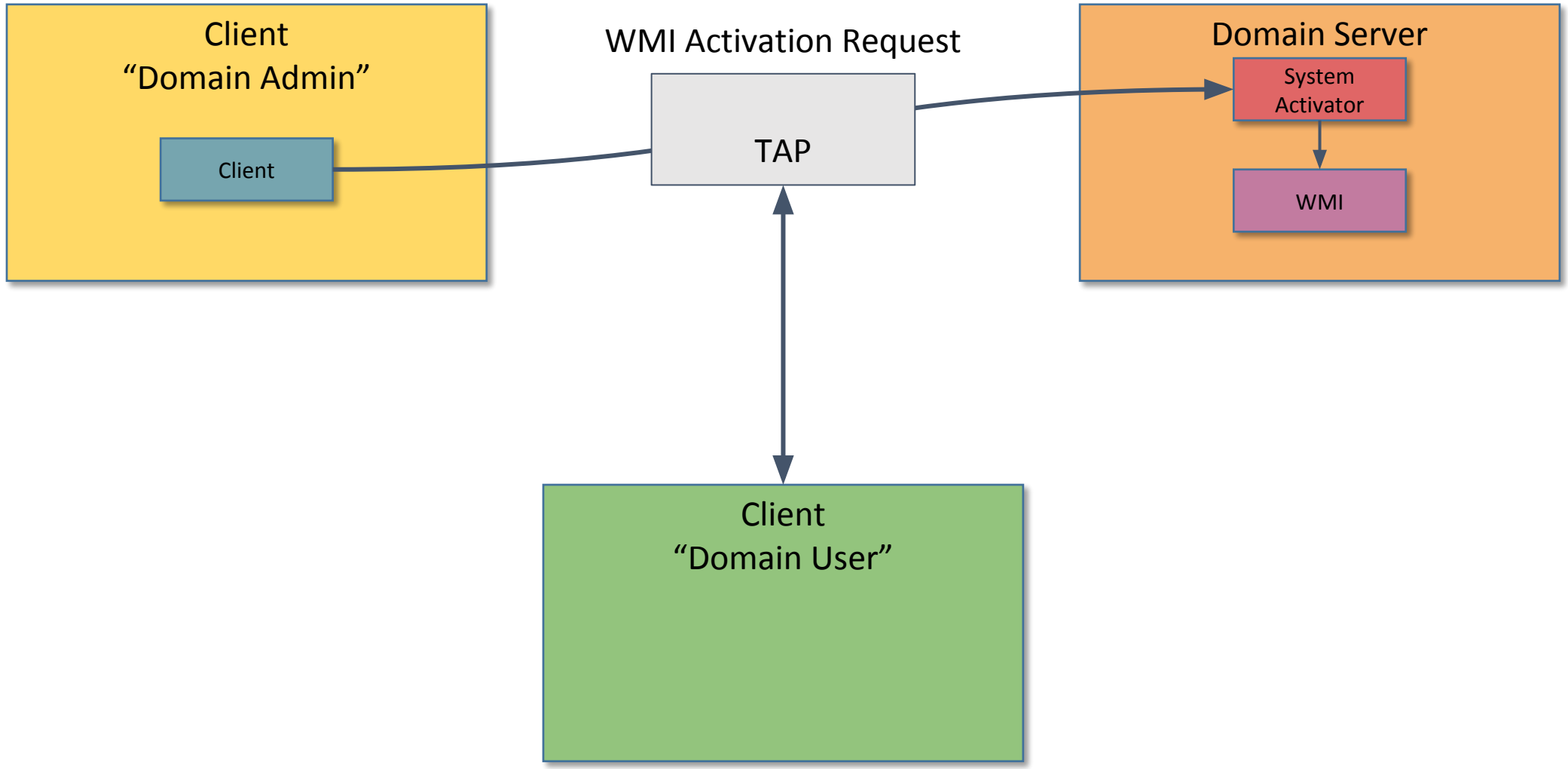


Access Permissions for WMI

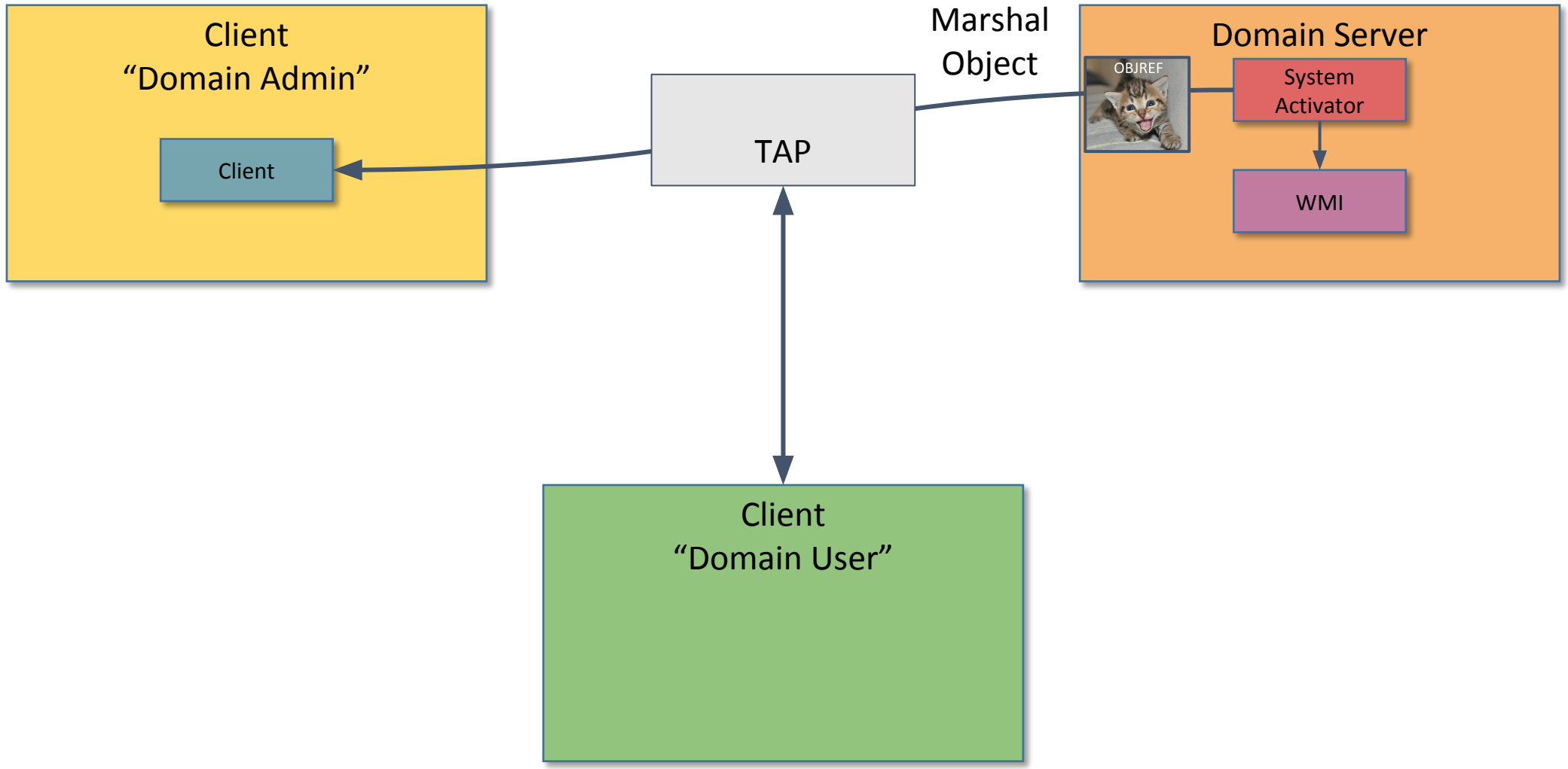


All authenticated users can access WMI remotely

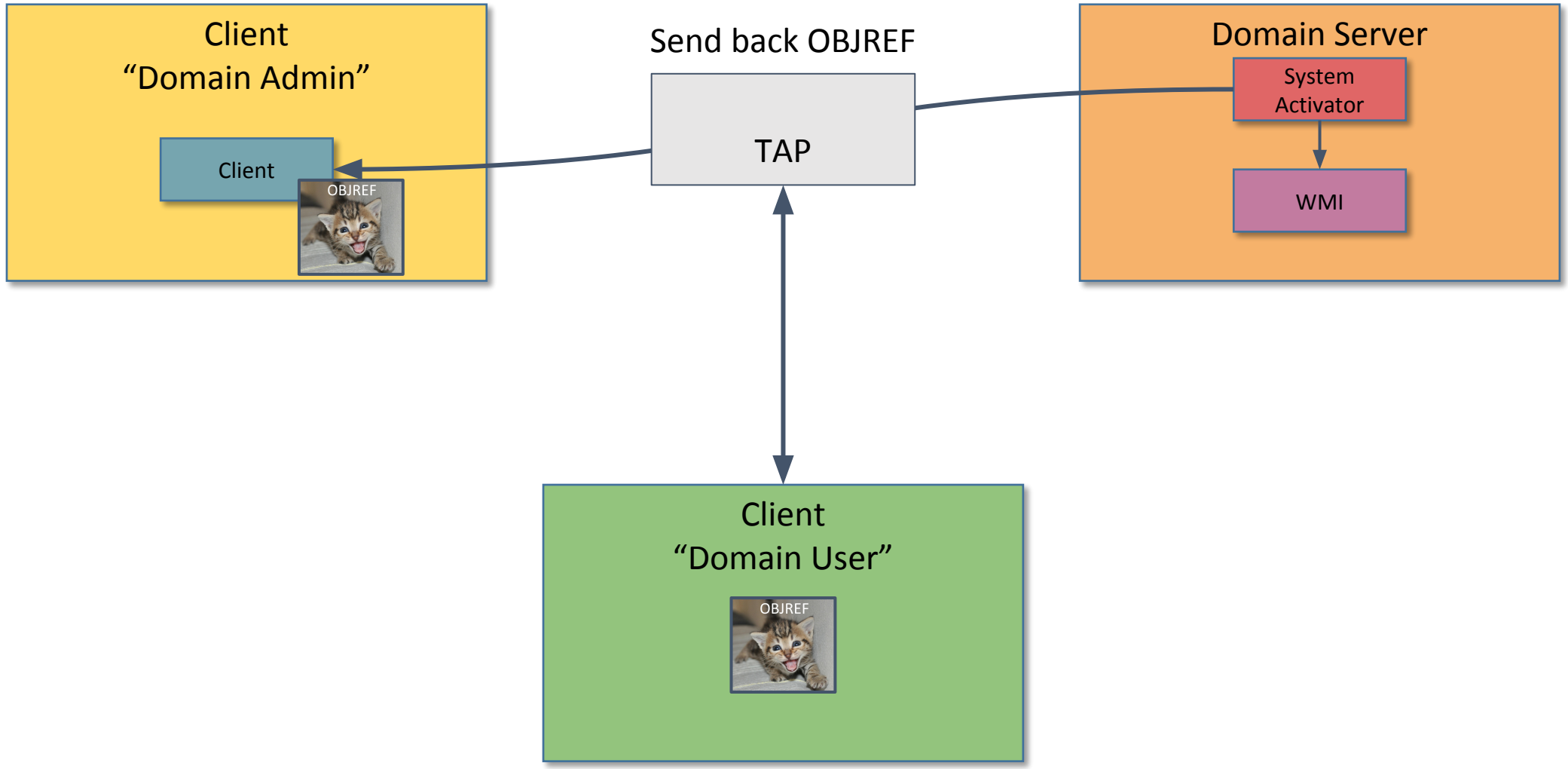
MEOW-Jacking!



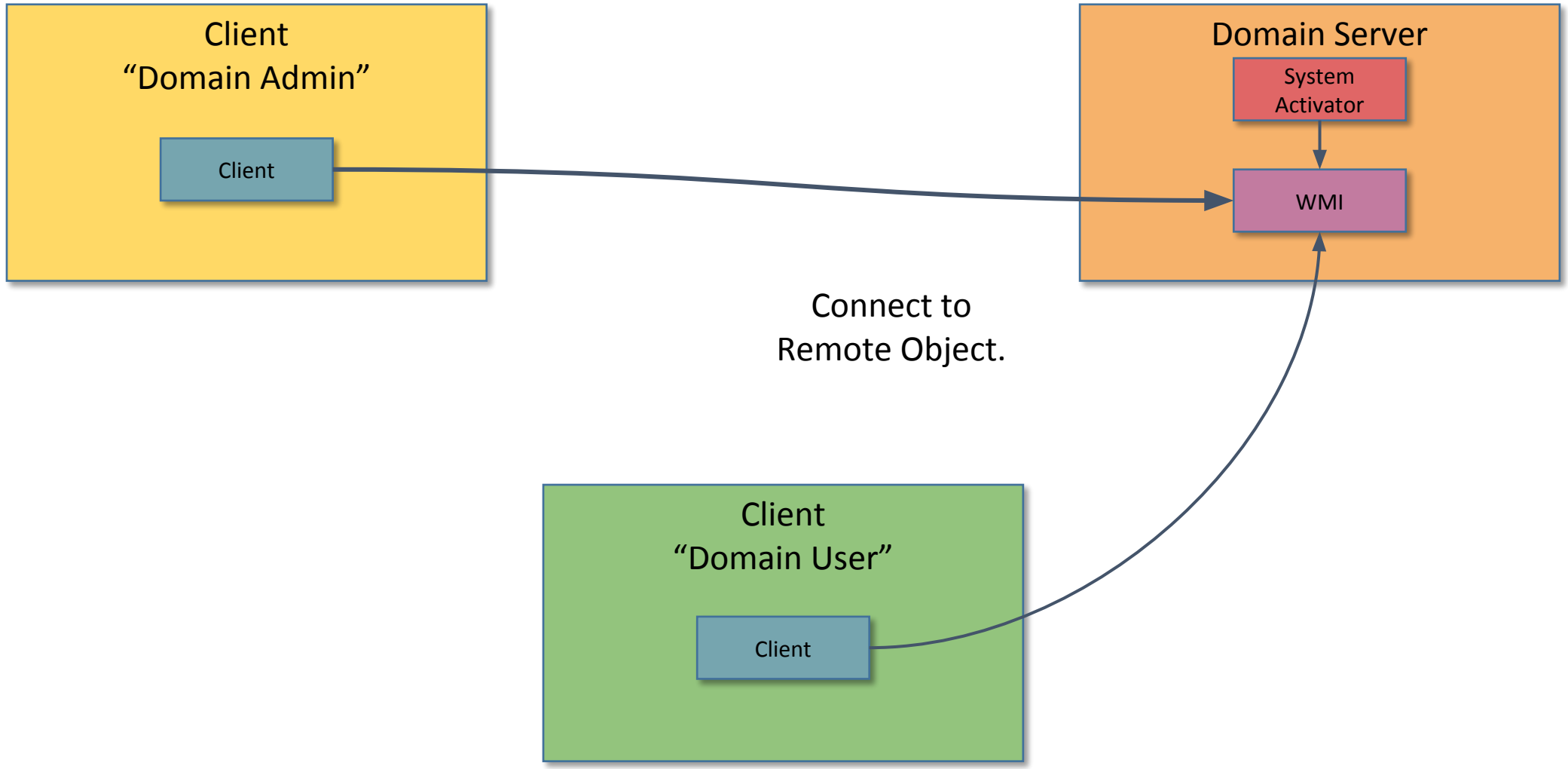
MEOW-Jacking!



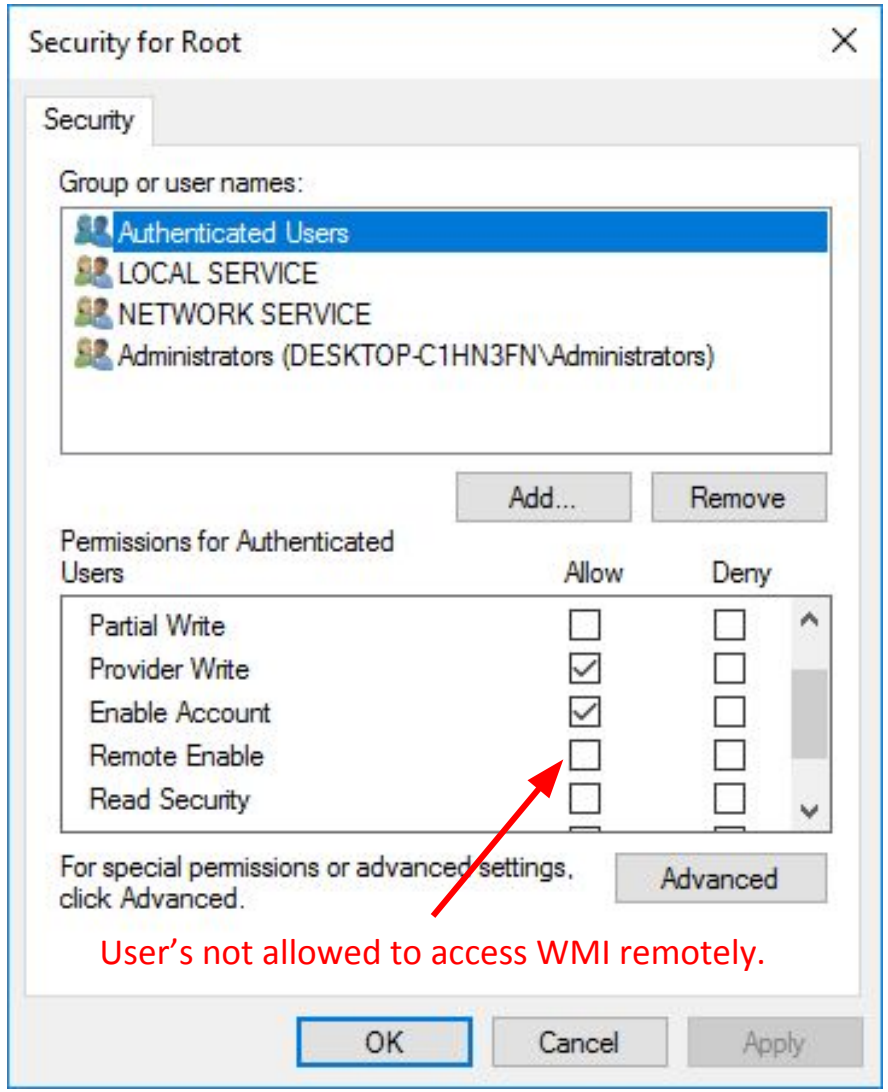
MEOW-Jacking!

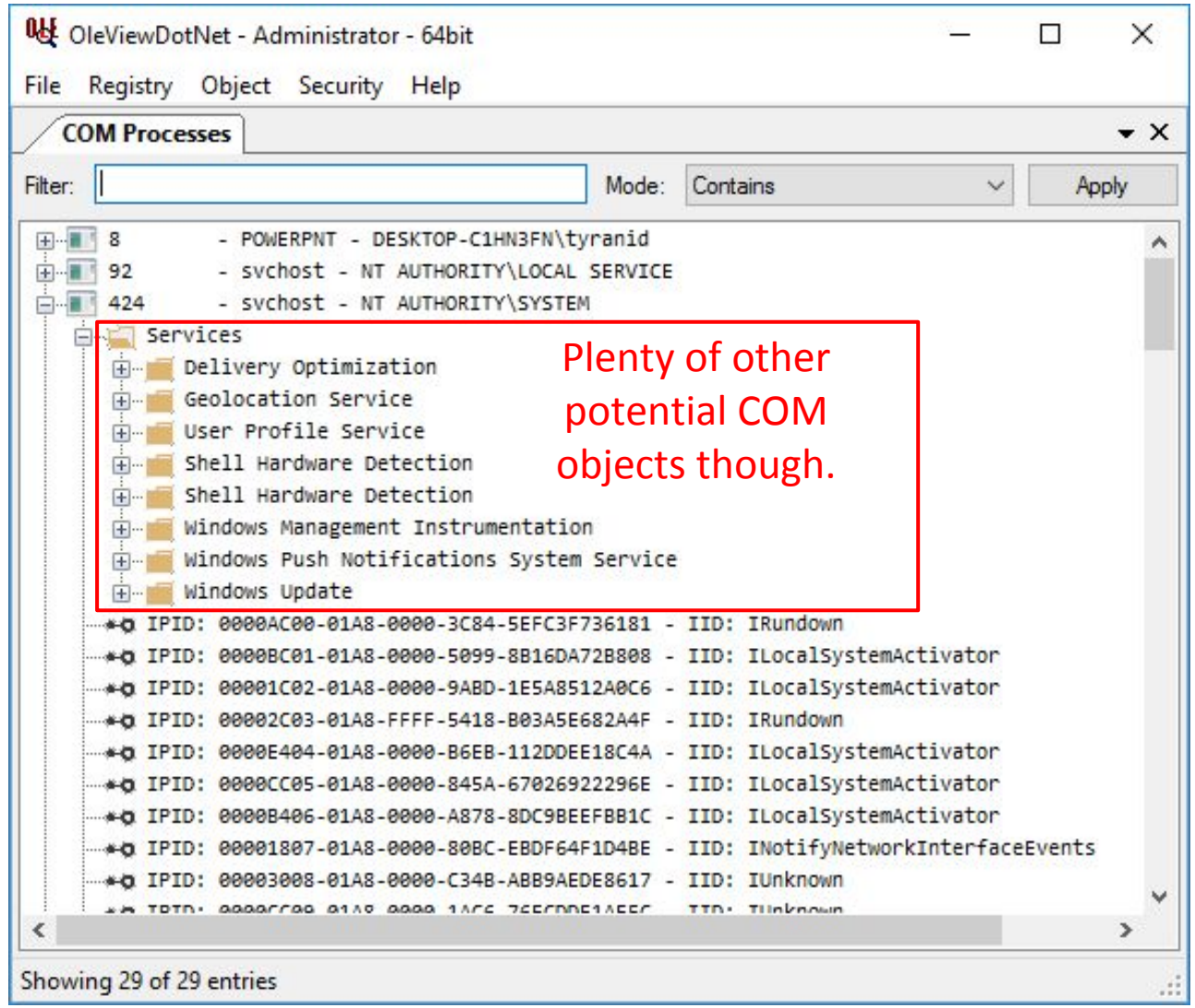


MEOW-Jacking!



Not as Bad as it Could Be





https://pixabay.com/en/pumpkin-halloween-frankenweenie-1789832/

https://pixabay.com/en/question-mark-pile-question-mark-1495858/