Demystifying COM

James Forshaw (@tiraniddo)

Troopers 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://github.com/tyranid/oleviewdotnet.binaries
https://goo.gl/Bkhlos

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

COM is Scary!

“Any sufficiently complex middleware is indistinguishable from magic.”

Arthur C. Clarke’s Third Law of Software Development

Mostly Documented'ish

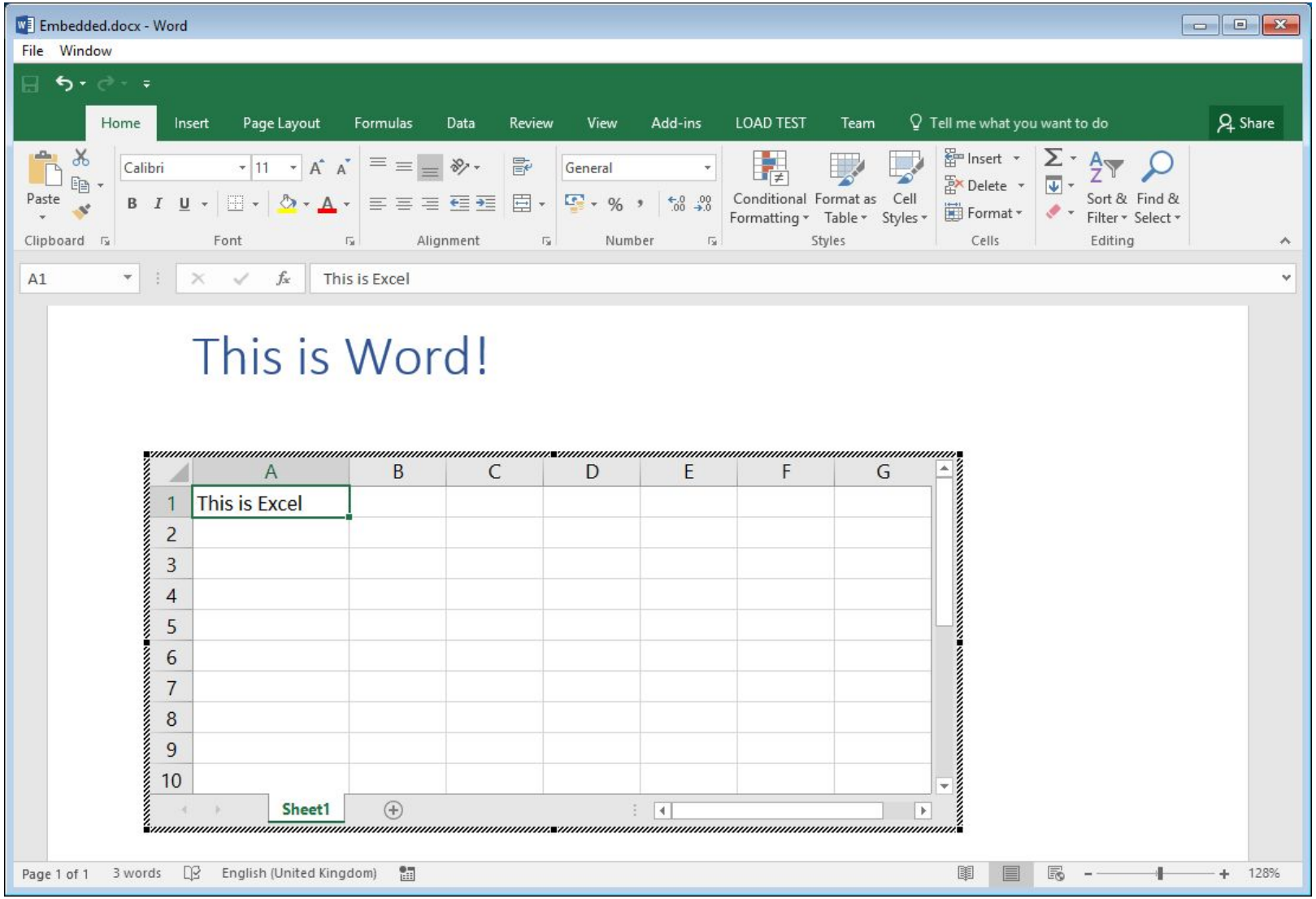
[MS-DCOM]:

Distributed Component Object Model (DCOM) Remote Protocol

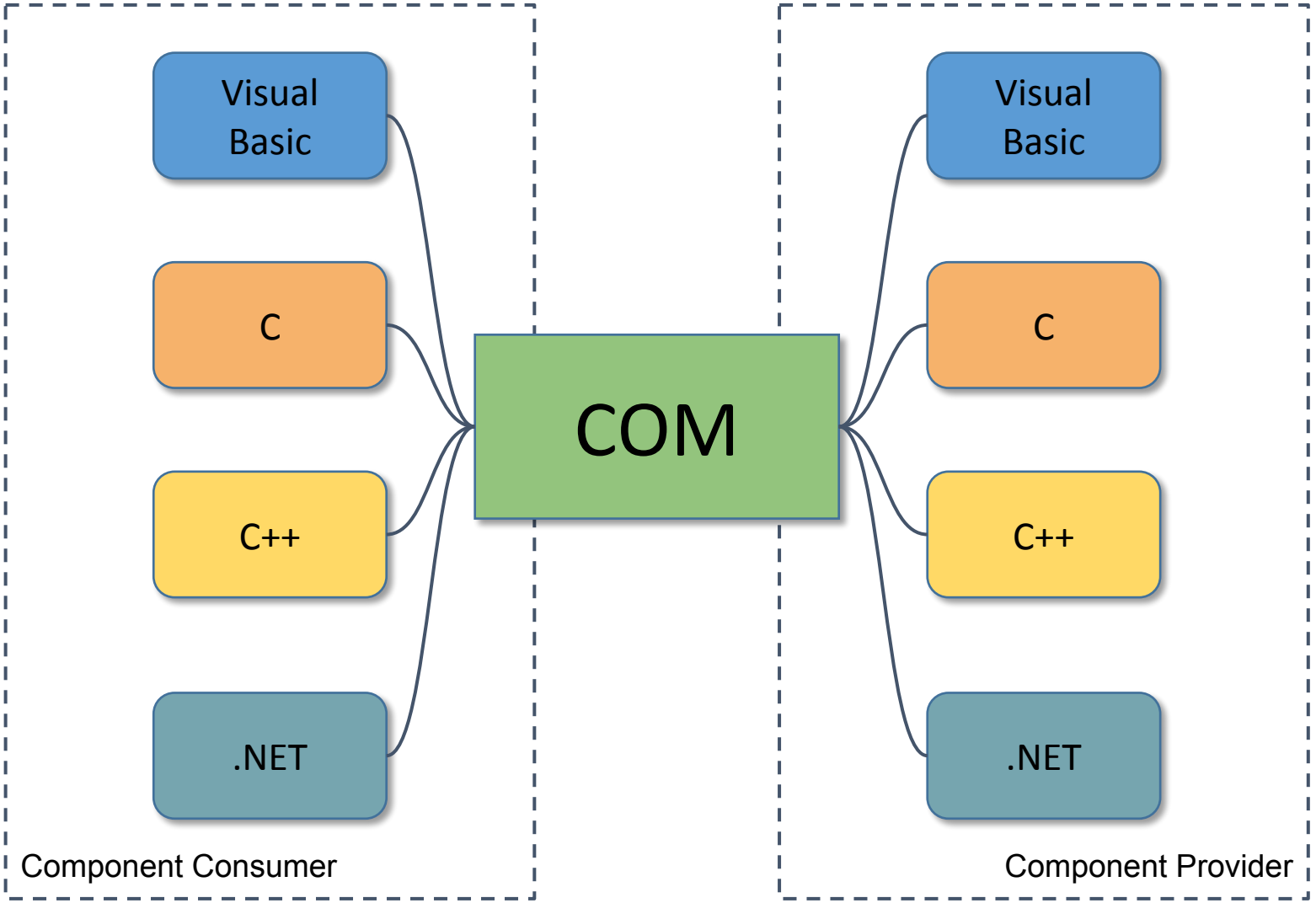
Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.

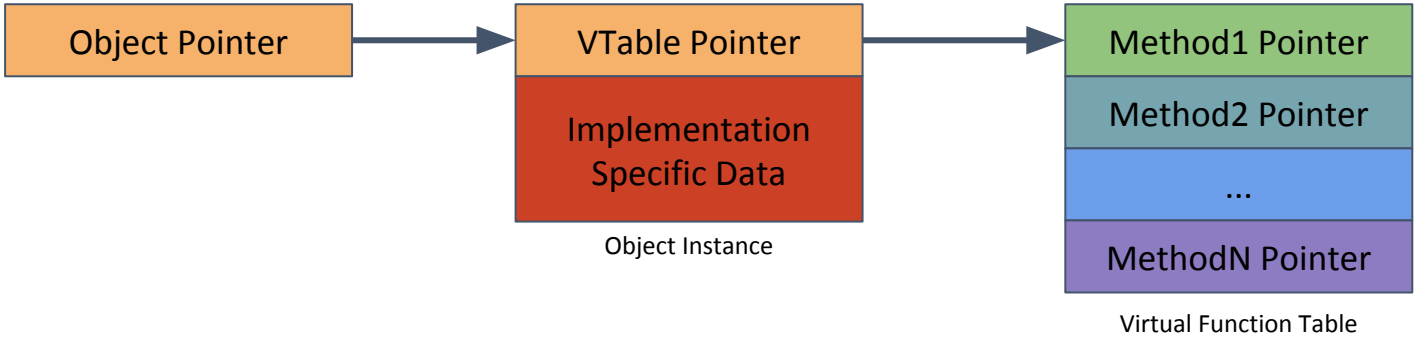
In the Beginning was OLE



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

C++ Implementation

```
struct Interface {
  virtual void SetInt(int i) = 0;
  virtual int GetInt() = 0;
};

class Object : public Interface {
 public:
  void SetInt(int i) {}
  int GetInt() {}
};

Interface* obj = new Object;
obj->SetInt(1234);
```

Define a pure virtual “Interface”

Derive from Interface and implement

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() {}
};
```

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

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();
};
```

Standard COM error code. >= 0 is Success

Used to reference count the object.

Interfaces defined using a 128 bit Globally Unique ID.

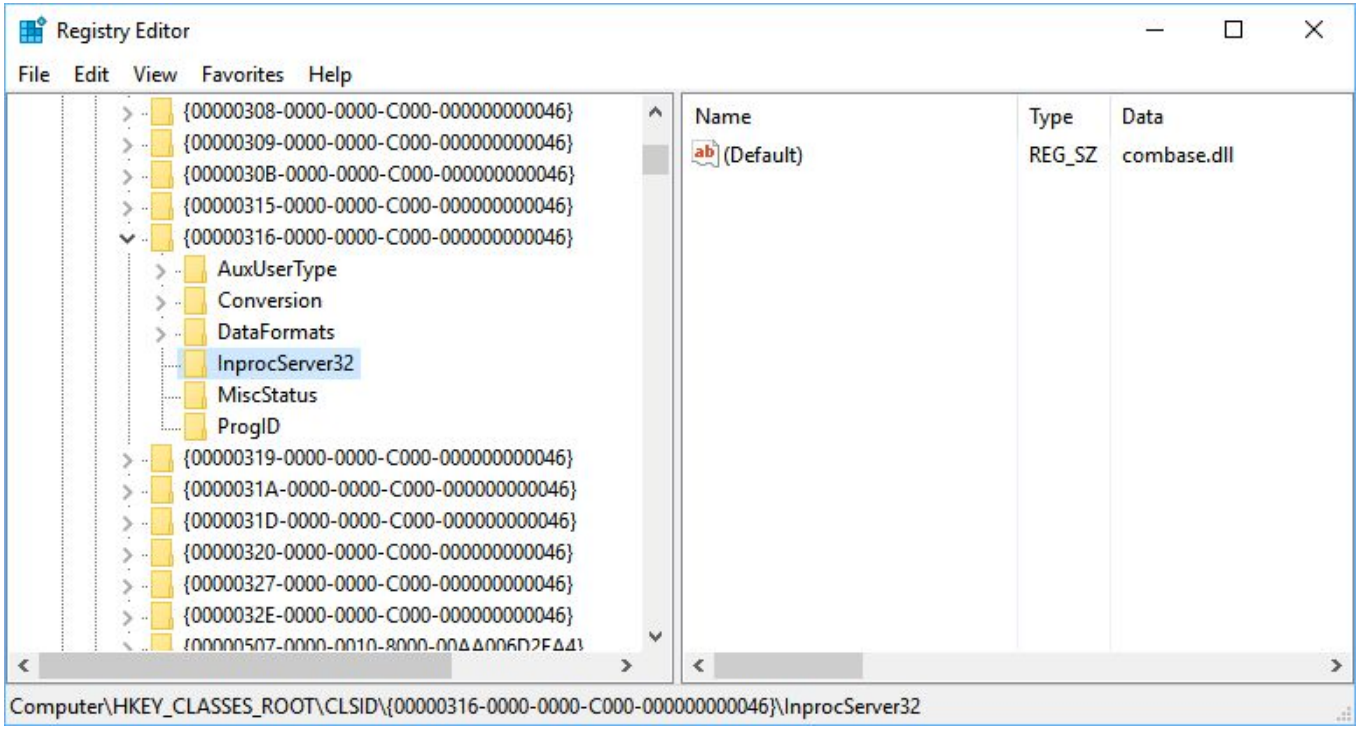
```
struct Interface1 : public IUnknown {};
struct Interface2 : public IUnknown {};

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

- Each Class Factory has a unique GUID, the Class ID (CLSID)
- Classes are registered in the Windows Registry under HKEY_CLASSES_ROOT\CLSID\{GUID}
- Registration information depends on the type of server



Class Factories

- Component classes can't be directly 'newed' so COM defines a factory interface, *IClassFactory*

```
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);
};
```

Used for COM Aggregation. Best ignored :-)

Creating Class Factories and Instances

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

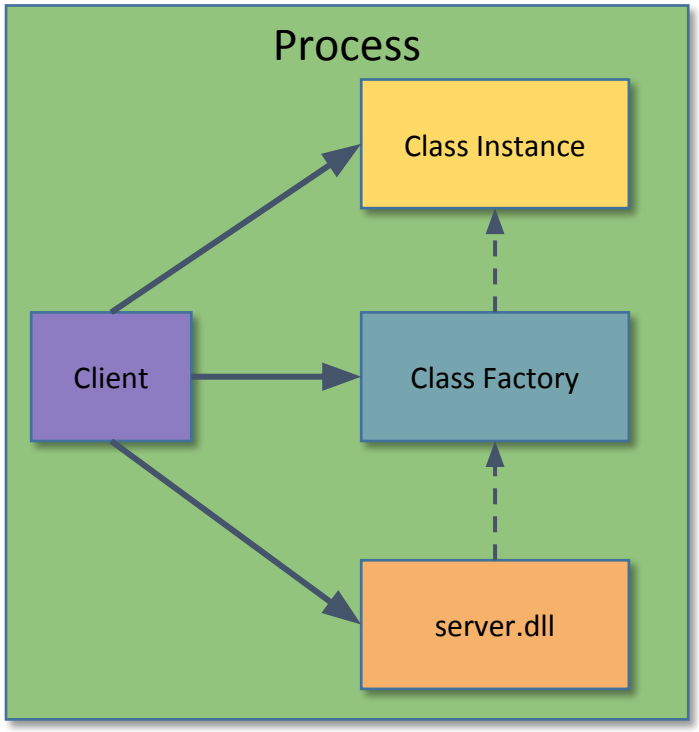
Specifies what type of server to lookup:

- 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).

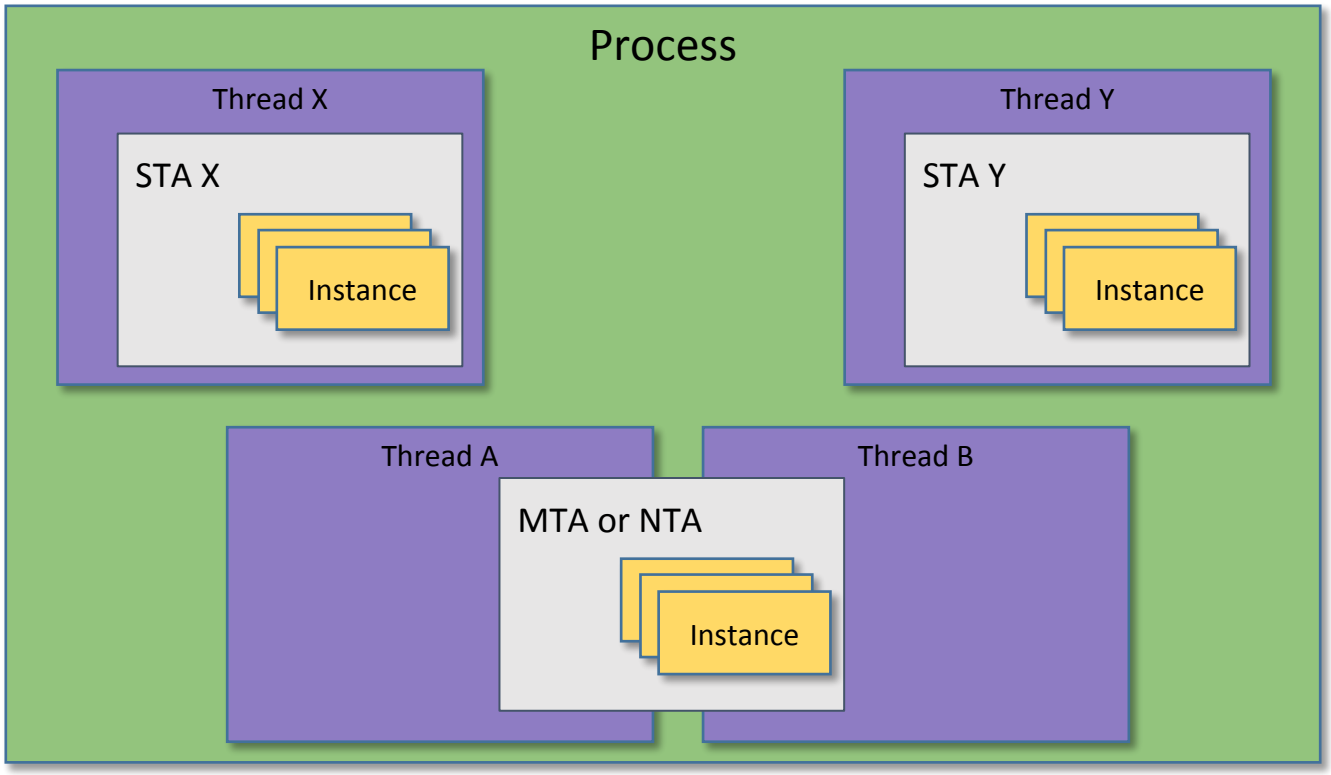
In-Process Server



- 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

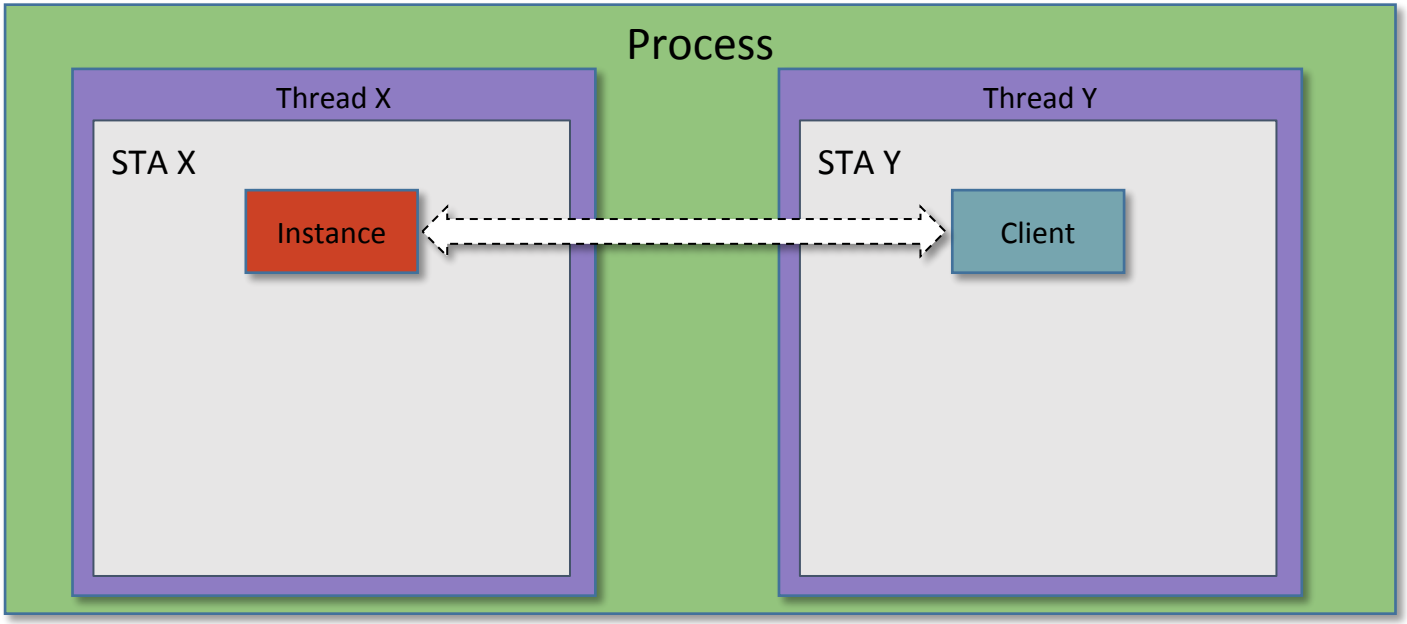



```
HRESULT CoInitializeEx(
  LPVOID pvReserved,
  DWORD  dwCoInit
);
```

```
COINIT_APARTMENTTHREADED
or
COINIT_MULTITHREADED
```

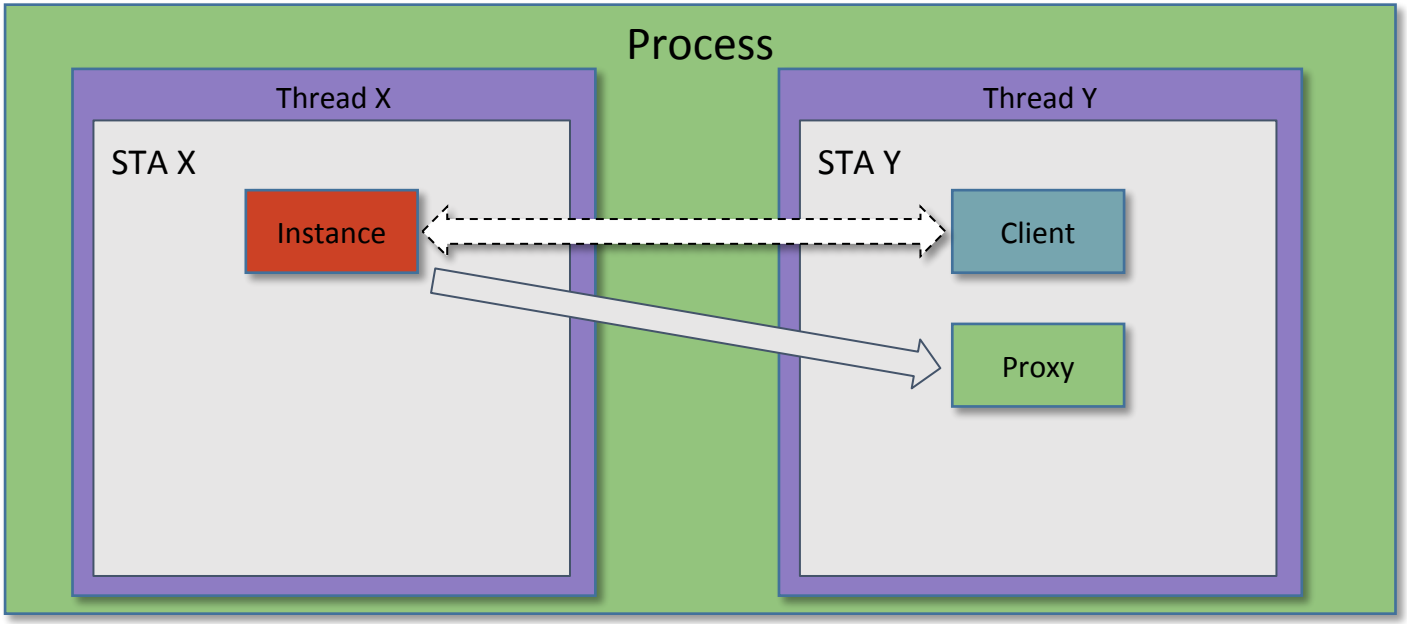
Single Threaded Apartments (STA)

- One STA per-thread. Initialized with COINIT_APARTMENTTHREADED.
- Created instance can only be called directly from the thread it was created on.
- Any other callers need to “Unmarshal” a Proxy to the object. Calls are sent to the original thread via a hidden Window and Window messaging



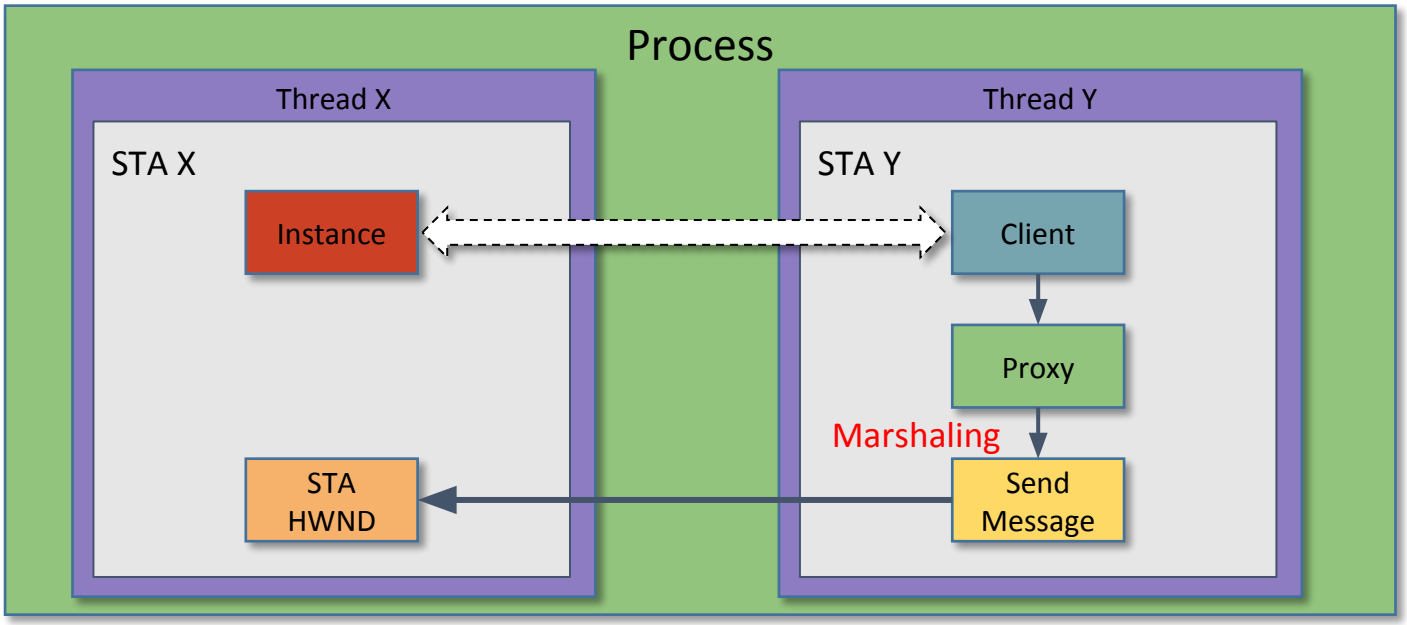
Single Threaded Apartments (STA)

- One STA per-thread. Initialized with COINIT_APARTMENTTHREADED.
- Created instance can only be called directly from the thread it was created on.
- Any other callers need to “Unmarshal” a Proxy to the object. Calls are sent to the original thread via a hidden Window and Window messaging



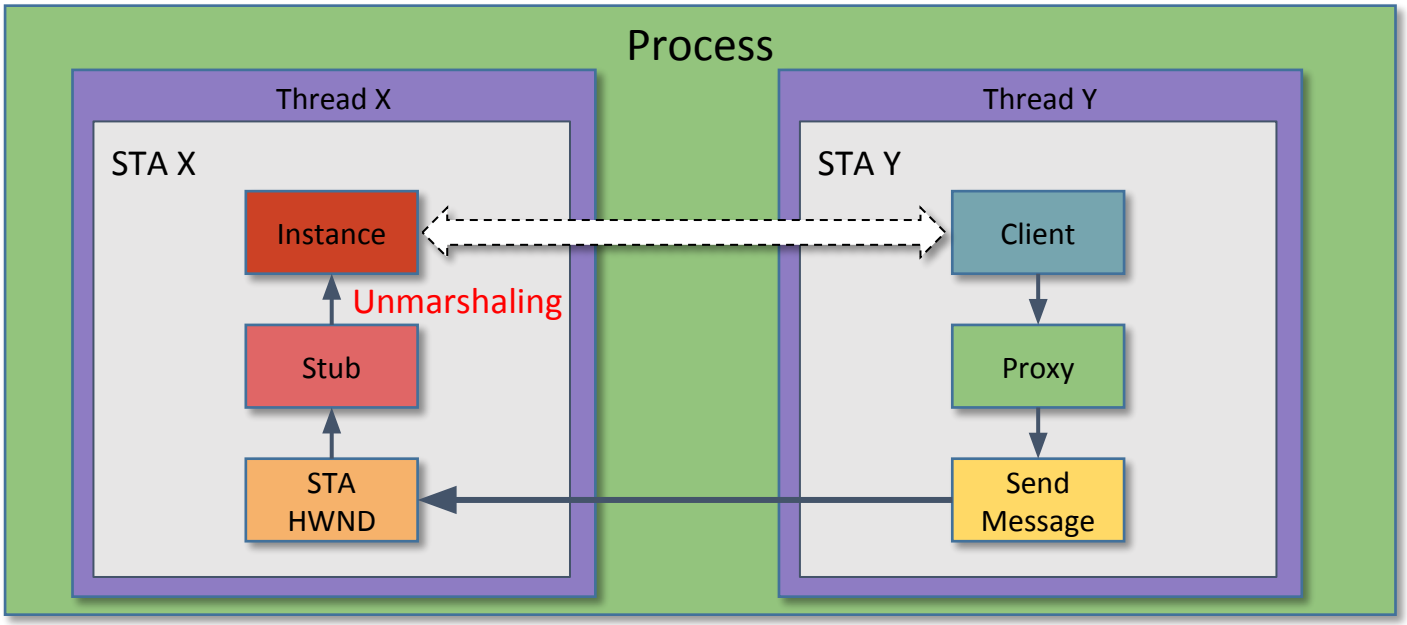
Single Threaded Apartments (STA)

- One STA per-thread. Initialized with COINIT_APARTMENTTHREADED.
- Created instance can only be called directly from the thread it was created on.
- Any other callers need to “Unmarshal” a Proxy to the object. Calls are sent to the original thread via a hidden Window and Window messaging



Single Threaded Apartments (STA)

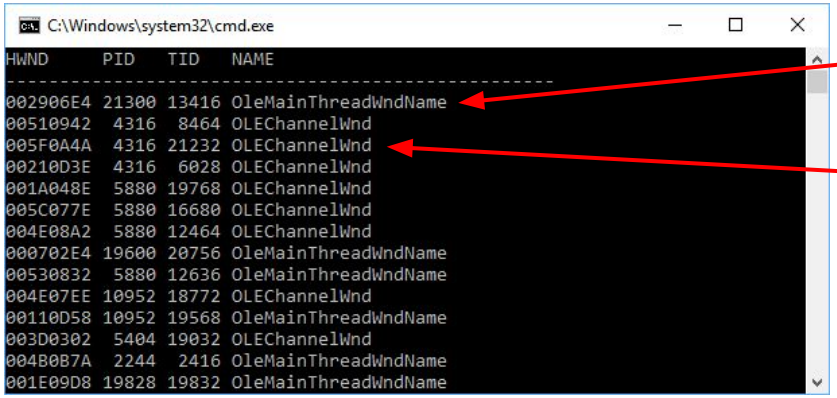
- One STA per-thread. Initialized with COINIT_APARTMENTTHREADED.
- Created instance can only be called directly from the thread it was created on.
- Any other callers need to “Unmarshal” a Proxy to the object. Calls are sent to the original thread via a hidden Window and Window messaging



The Mystery Window

- Initializing a STA creates a hidden “Message Only” 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);
}
```

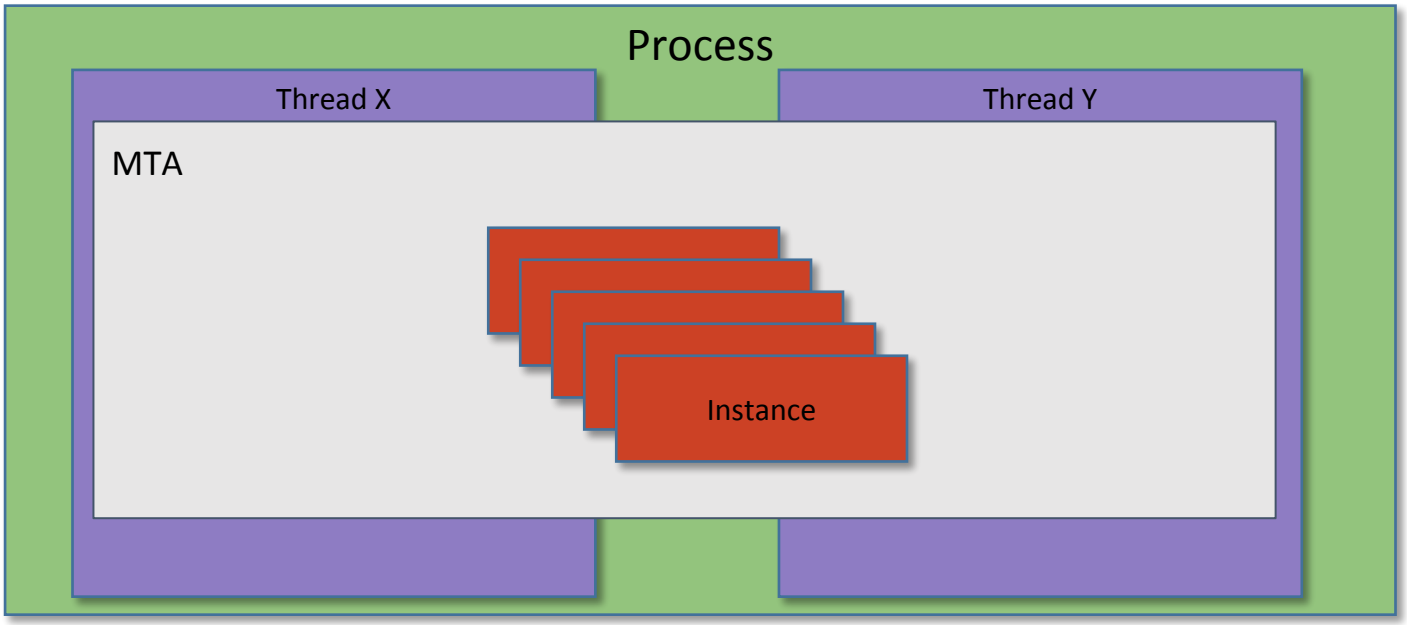


“Main” STA window.

Per-Thread STA window.

Multi-Threaded Apartments (MTA)

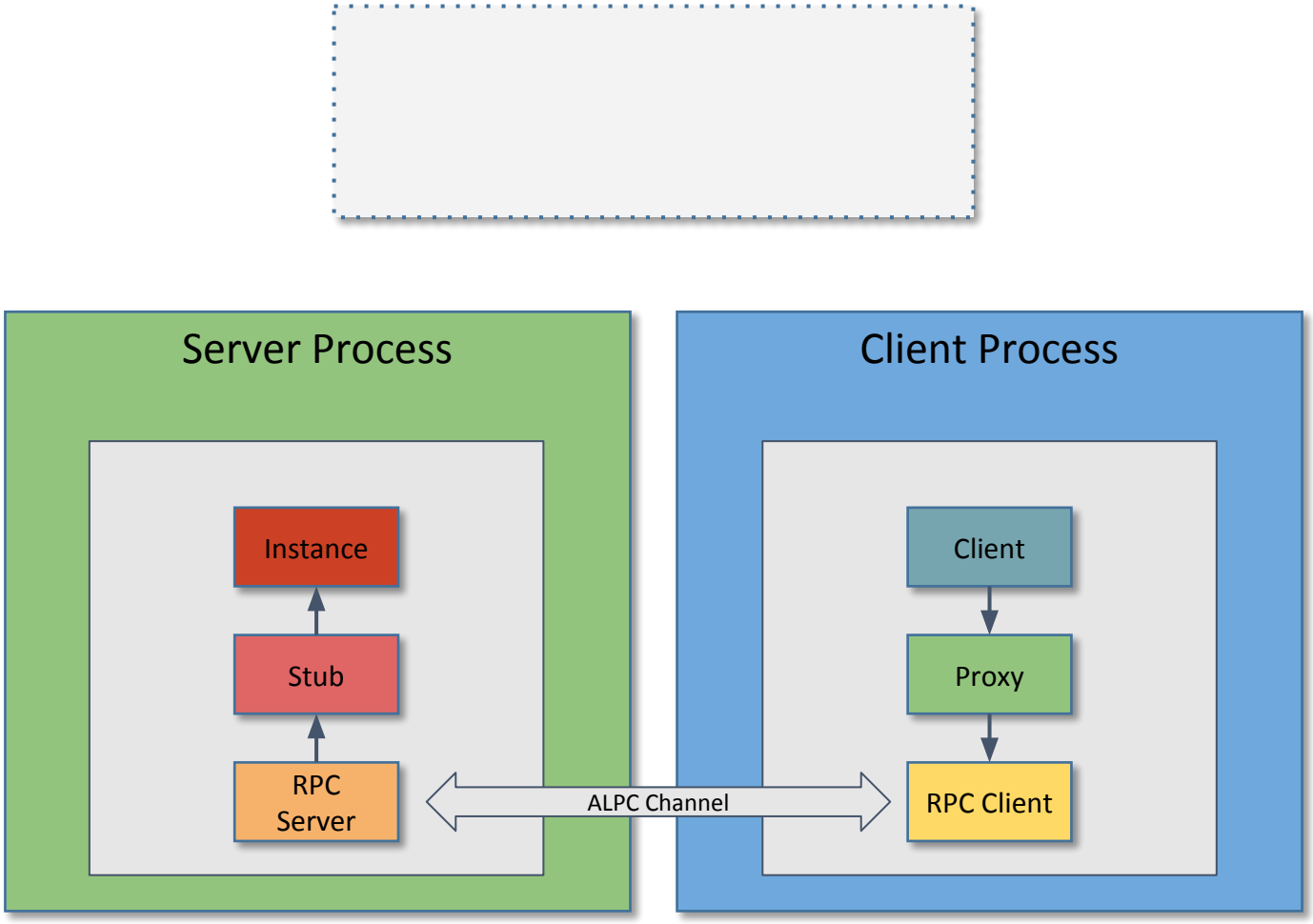
- One MTA per-process. Initialized with *COINIT_MULTITHREADED*.
- Created instance can be accessed directly in any thread in the MTA.
- STA instances need to be marshaled and vice-versa
- Also similar Neutral Apartment (NTA) which optimizes away some marshaling.



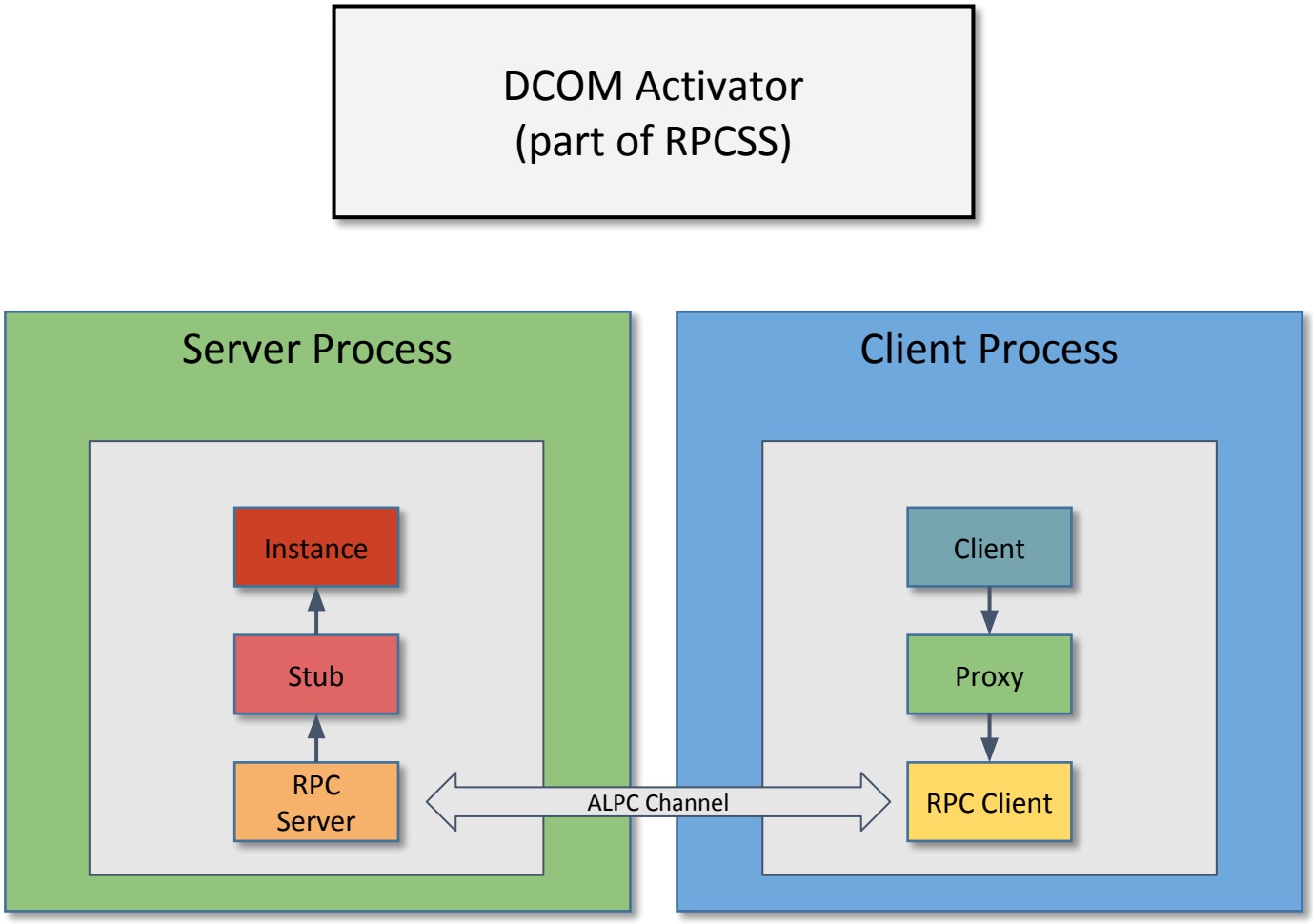
Threading Models

- In Process servers should be registered with a *ThreadingModel* registry value.
- Takes a string which must be one of the following values:
  - “Apartment” - Object must be created inside a STA
  - “Free” - Free-threaded, object must be created in the MTA
  - “Both” - Object can be created and used directly in either STA or MTA
  - “Neutral” - Object must be created in a NTA (COM+ Wizardry)
- If no *ThreadingModel* object can ONLY be used and created in the first STA in the process.
- If *Free* object created in a STA then MTA is created if it doesn’t exist and marshaled back.
- Likewise if *Apartment* object created in an MTA a new STA is created and object marshaled back

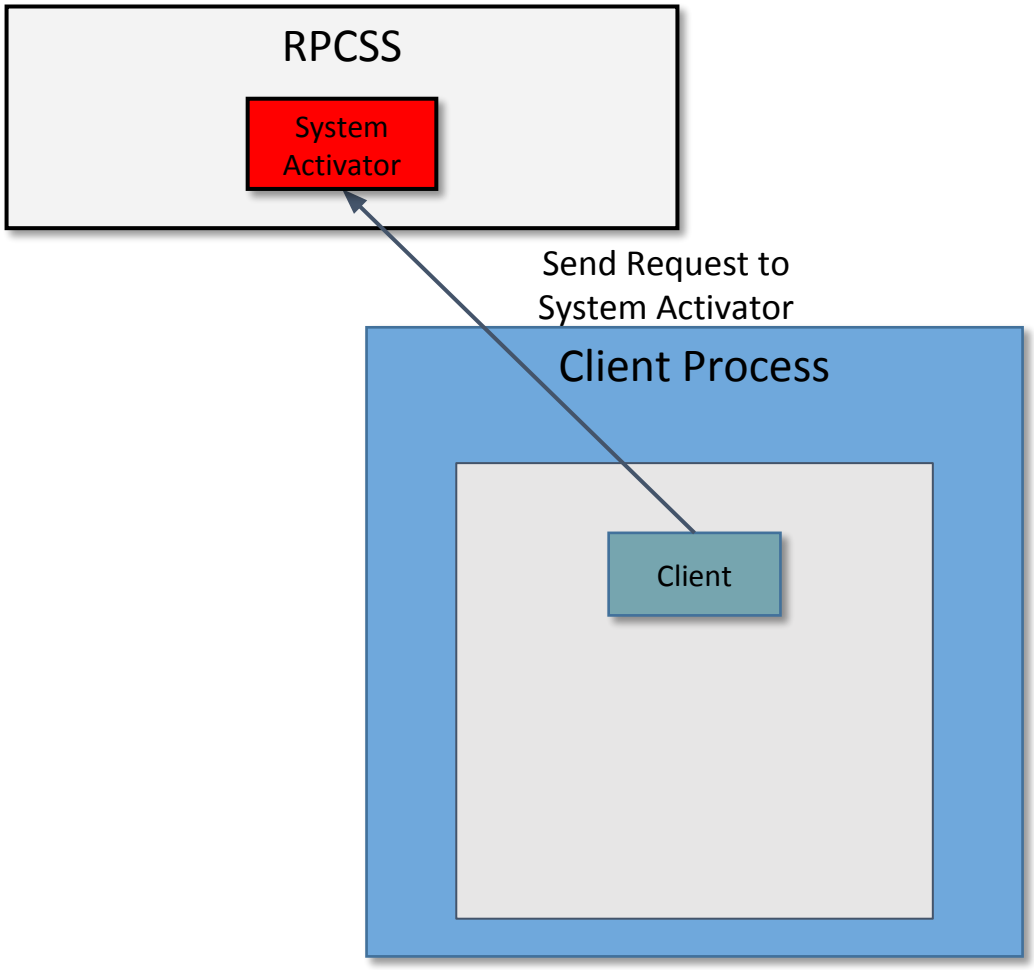
Local Server Activation



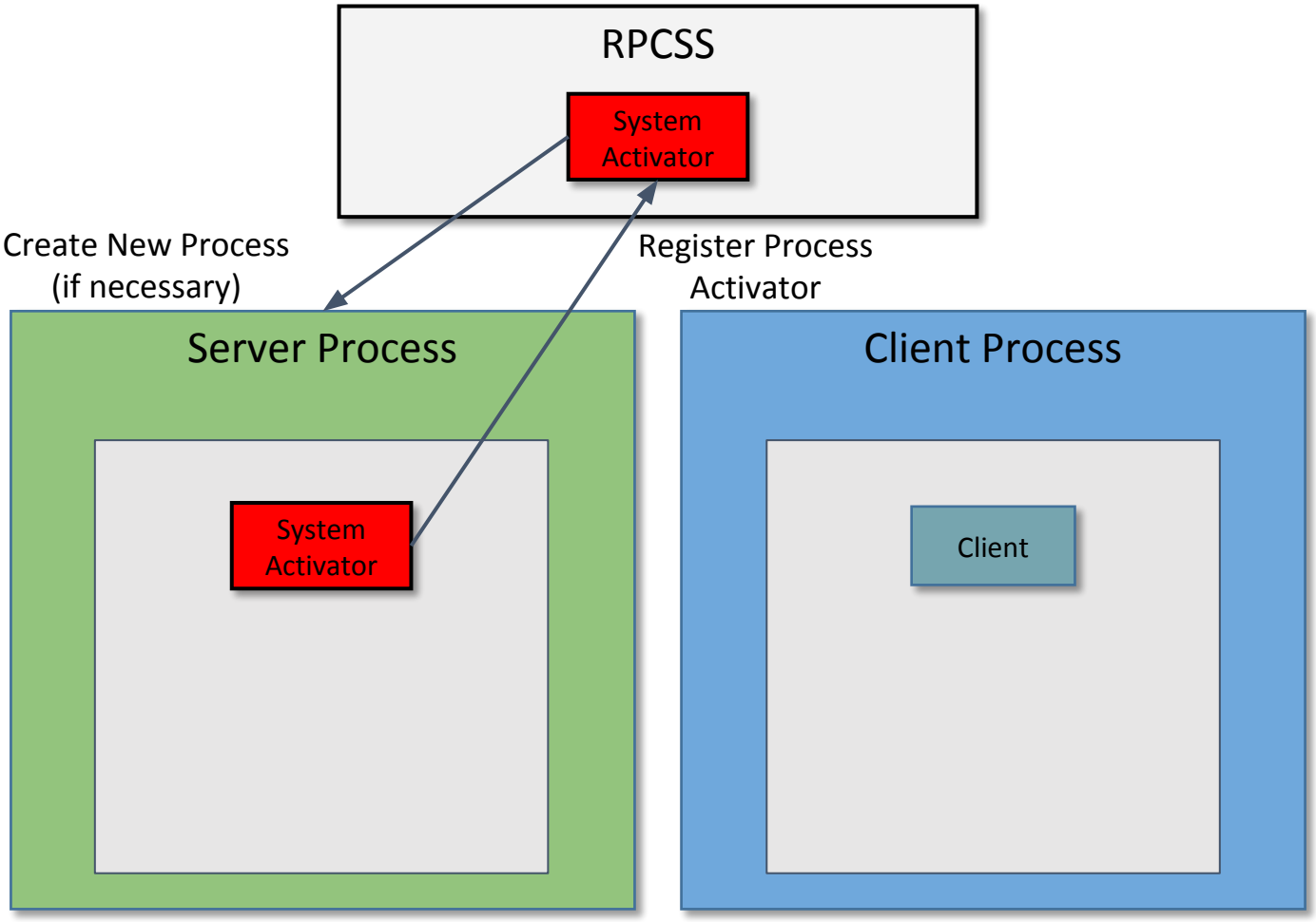
Local Server Activation



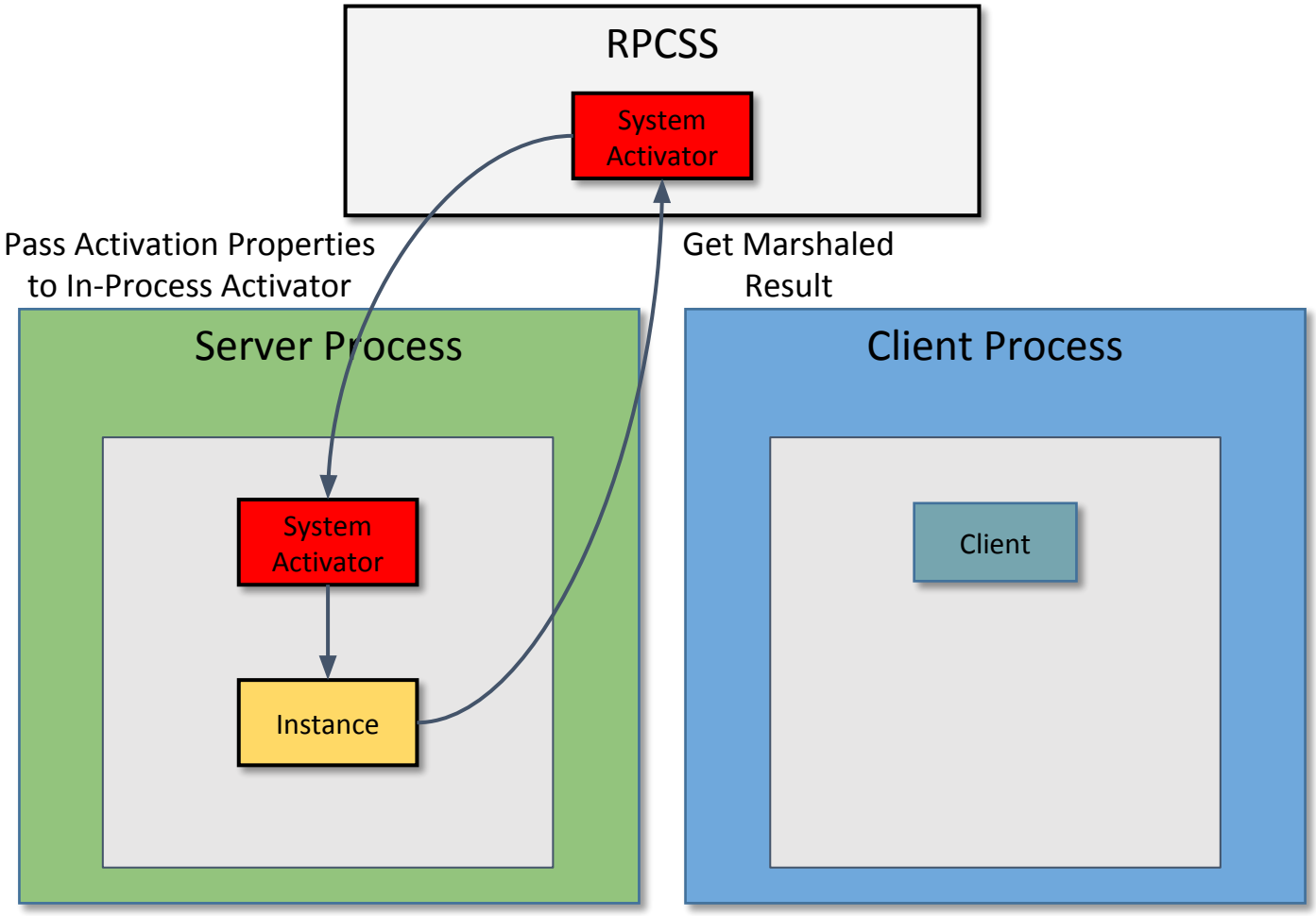
Local Server Activation



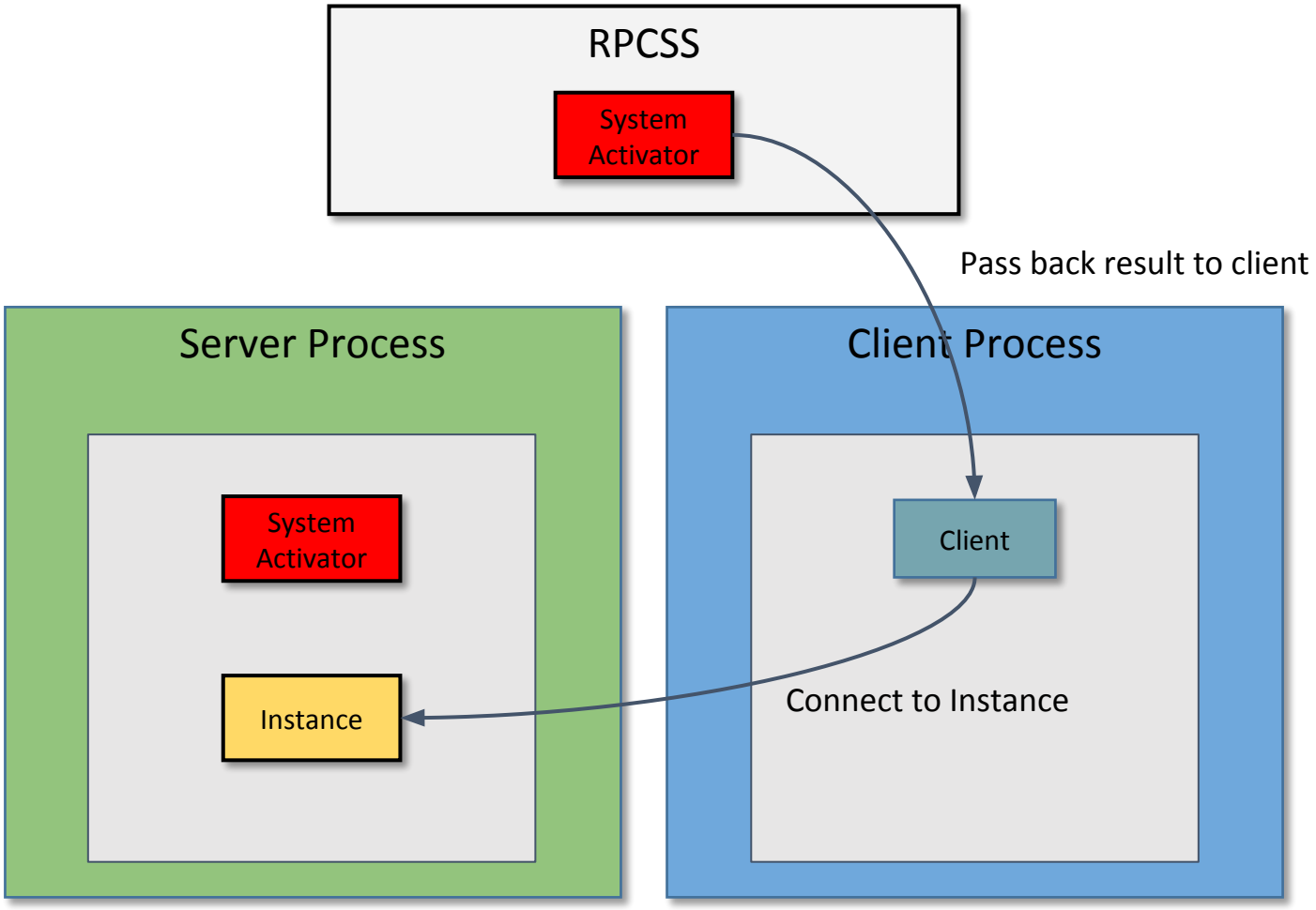
Local Server Activation



Local Server Activation



Local Server Activation

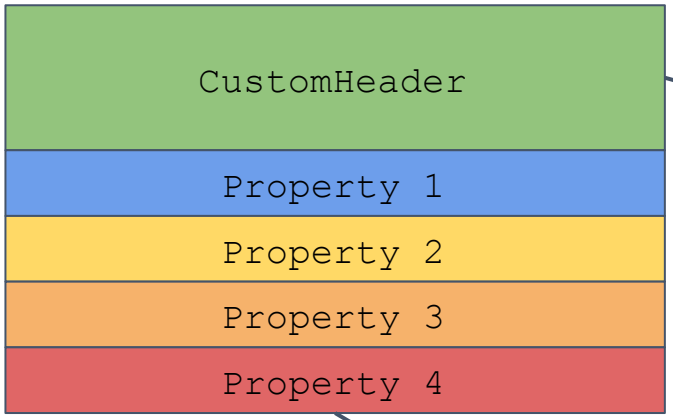


System Activator

```
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 of class 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

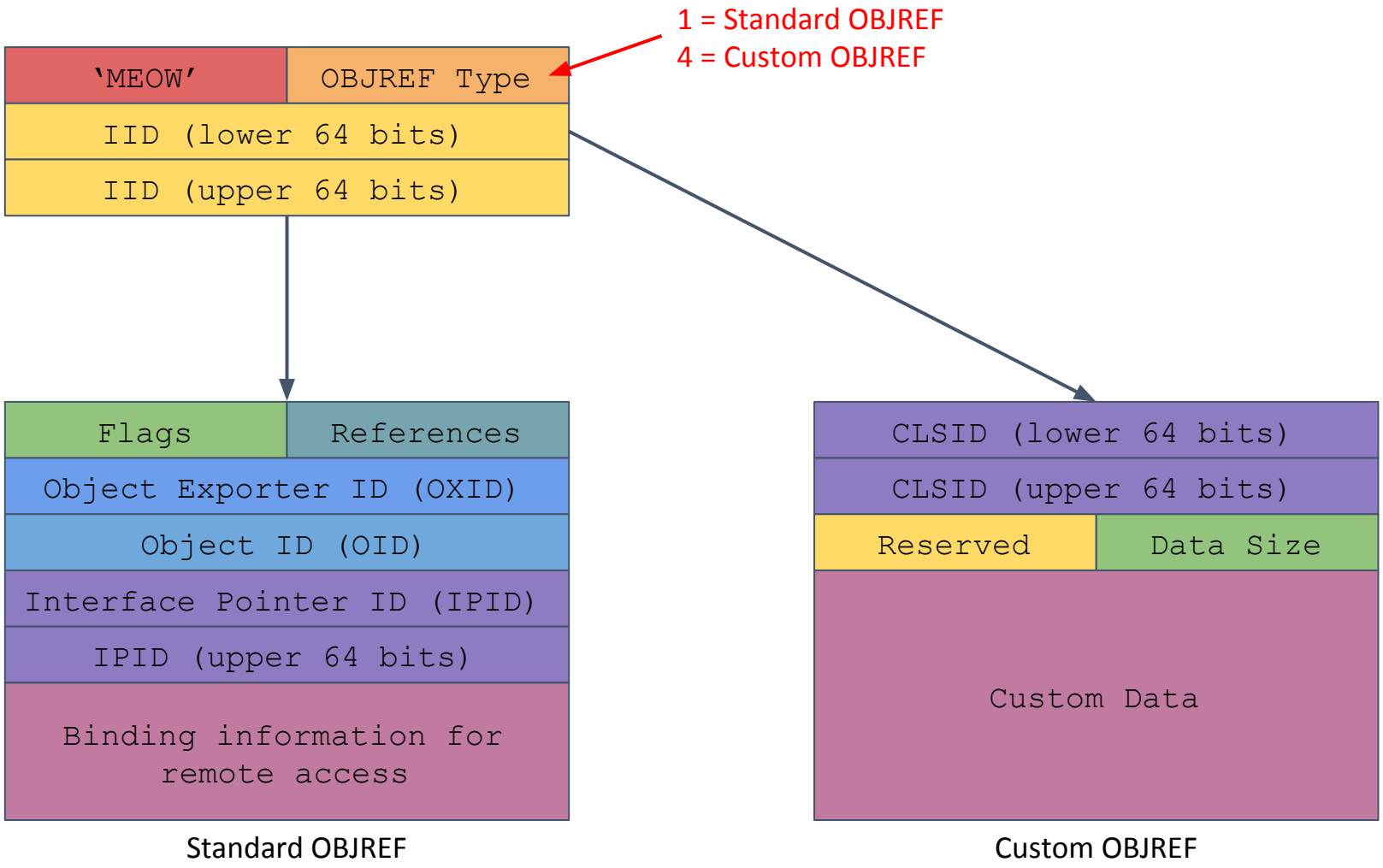
- Same structure as ActivationPropertiesIn just with different property sets.

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

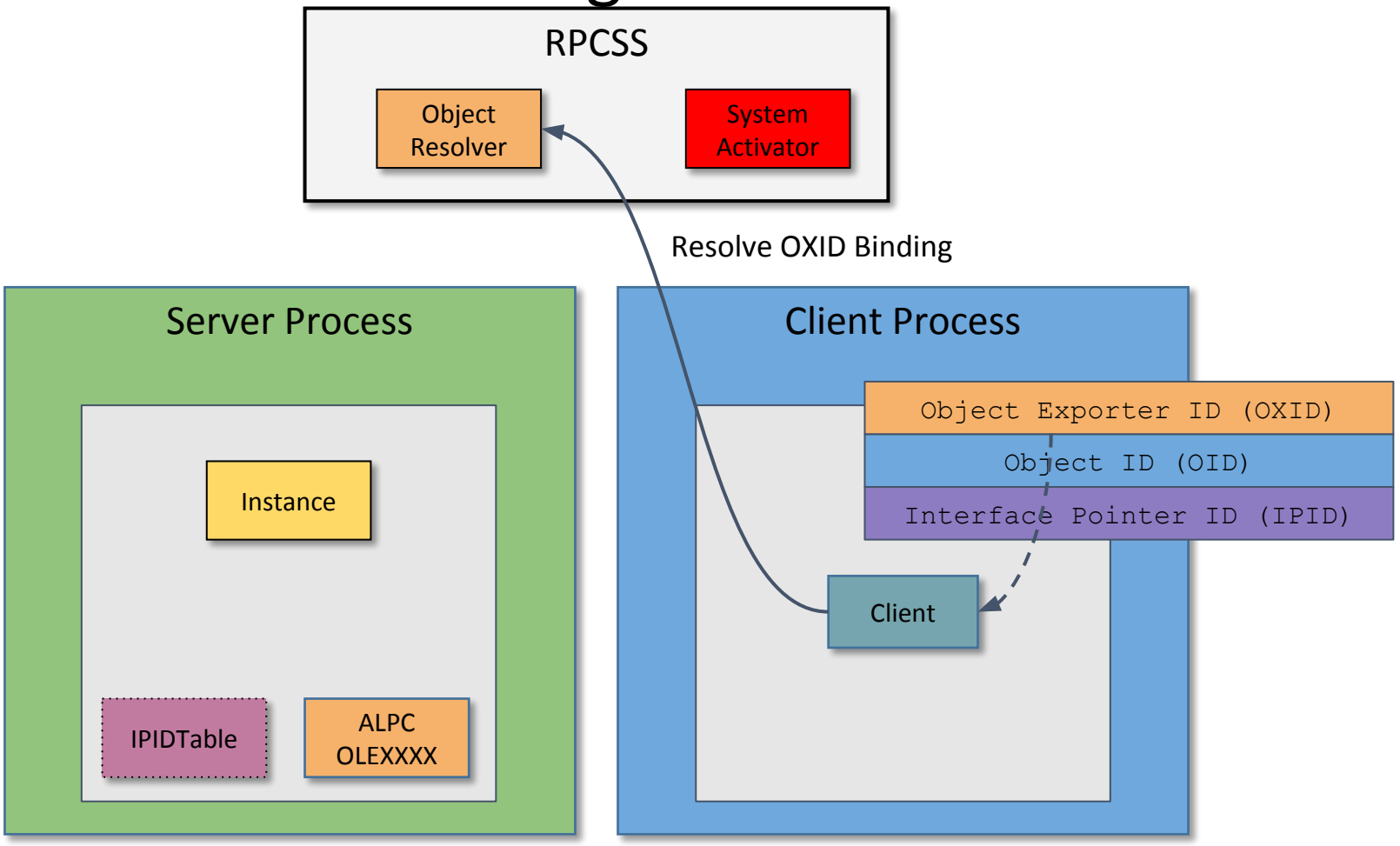
```
struct MInterfacePointer {
    unsigned long ulCntData;
    byte abData[];
};
```

```
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 ▮                                     ....
```

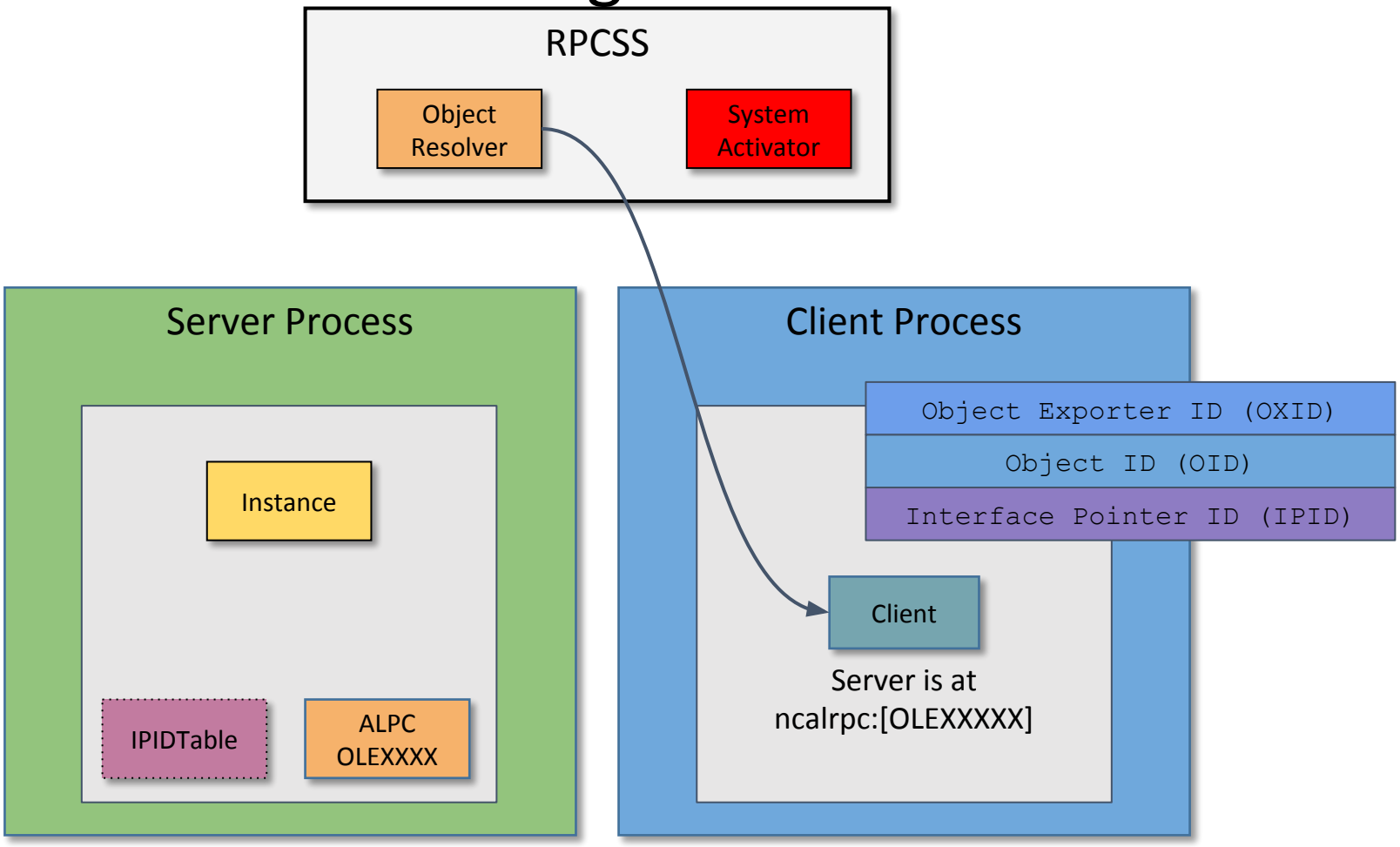
It’s like Marshaling Cats



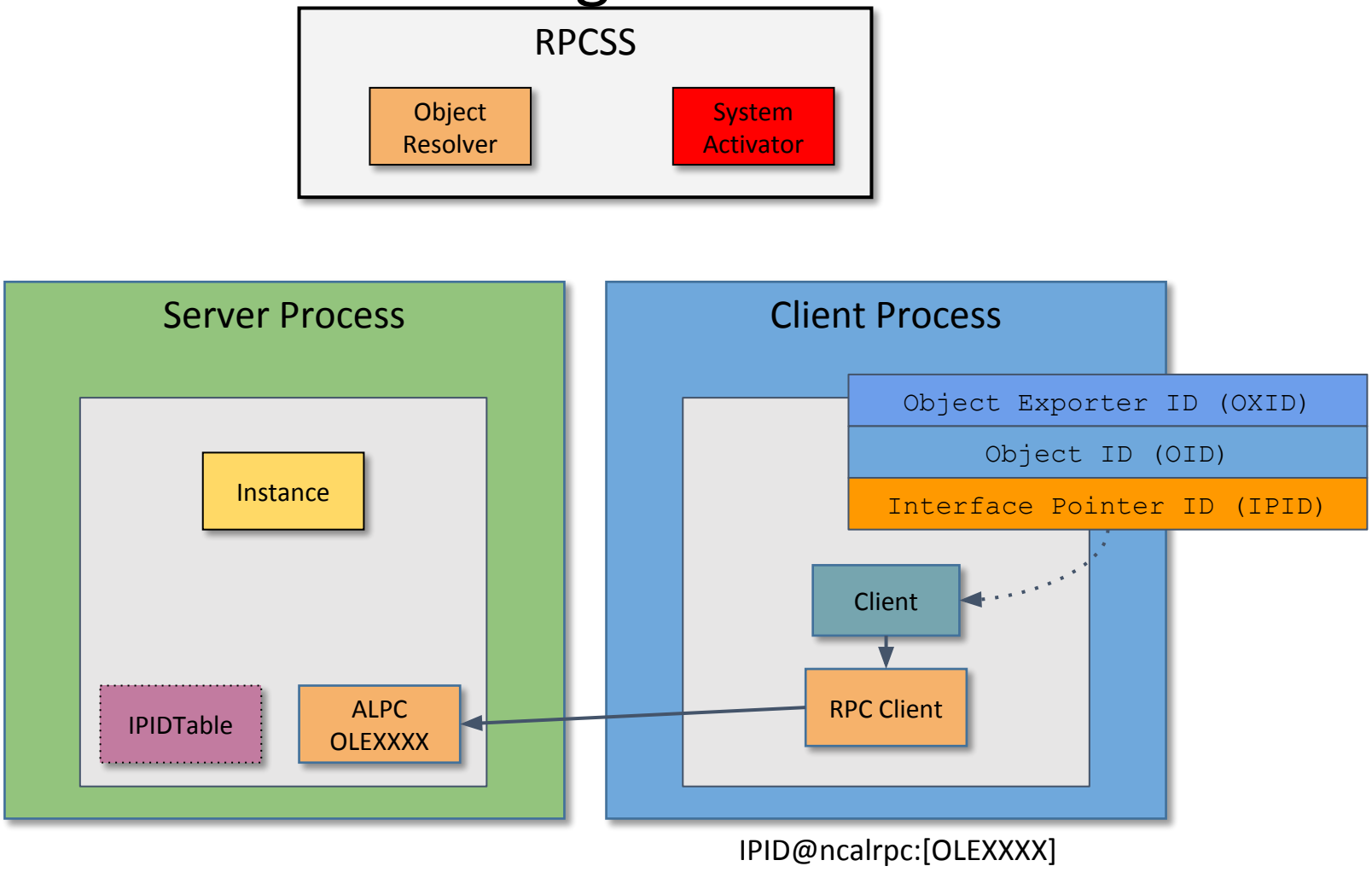
Standard Unmarshaling



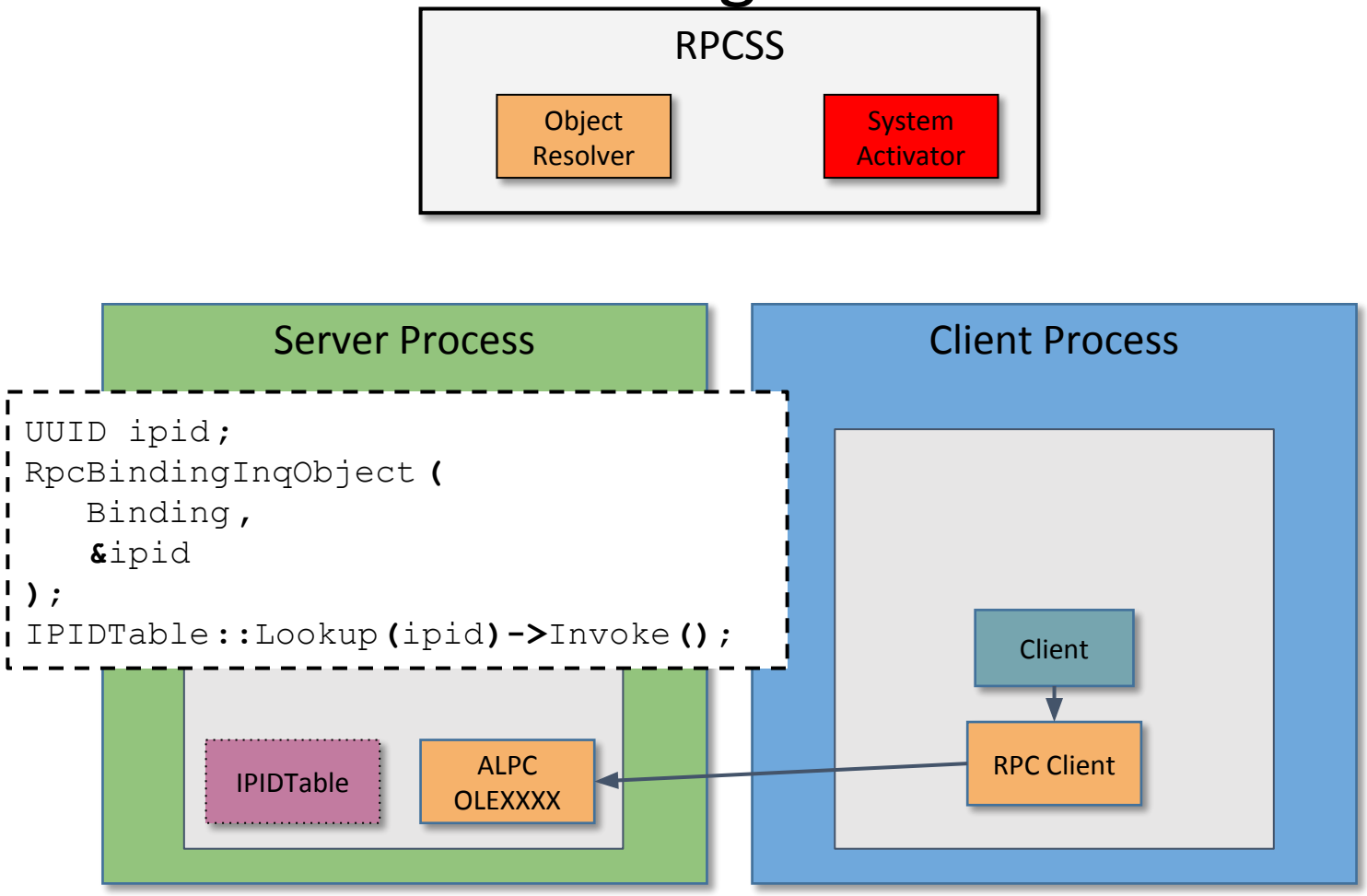
Standard Unmarshaling



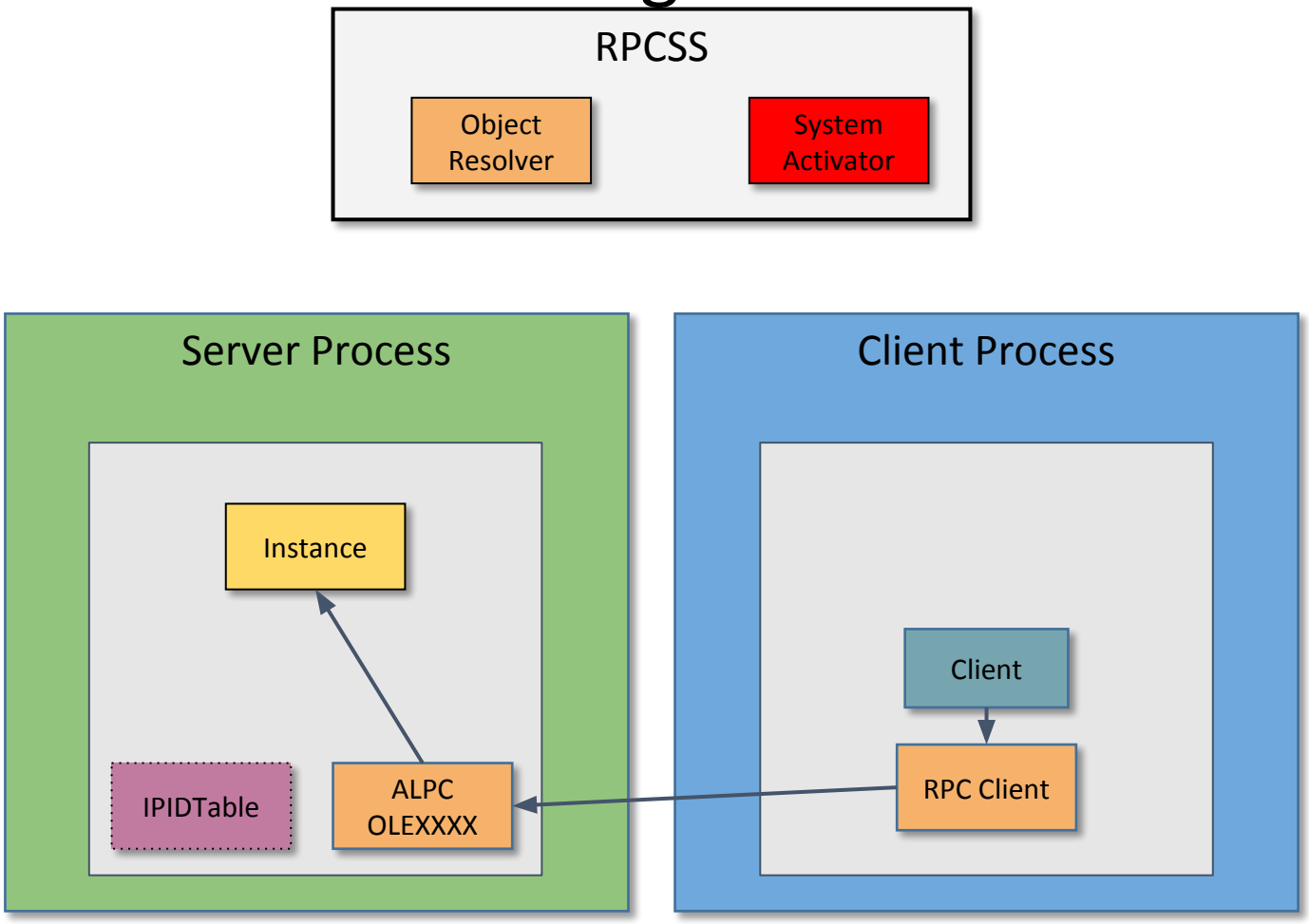
Standard Unmarshaling



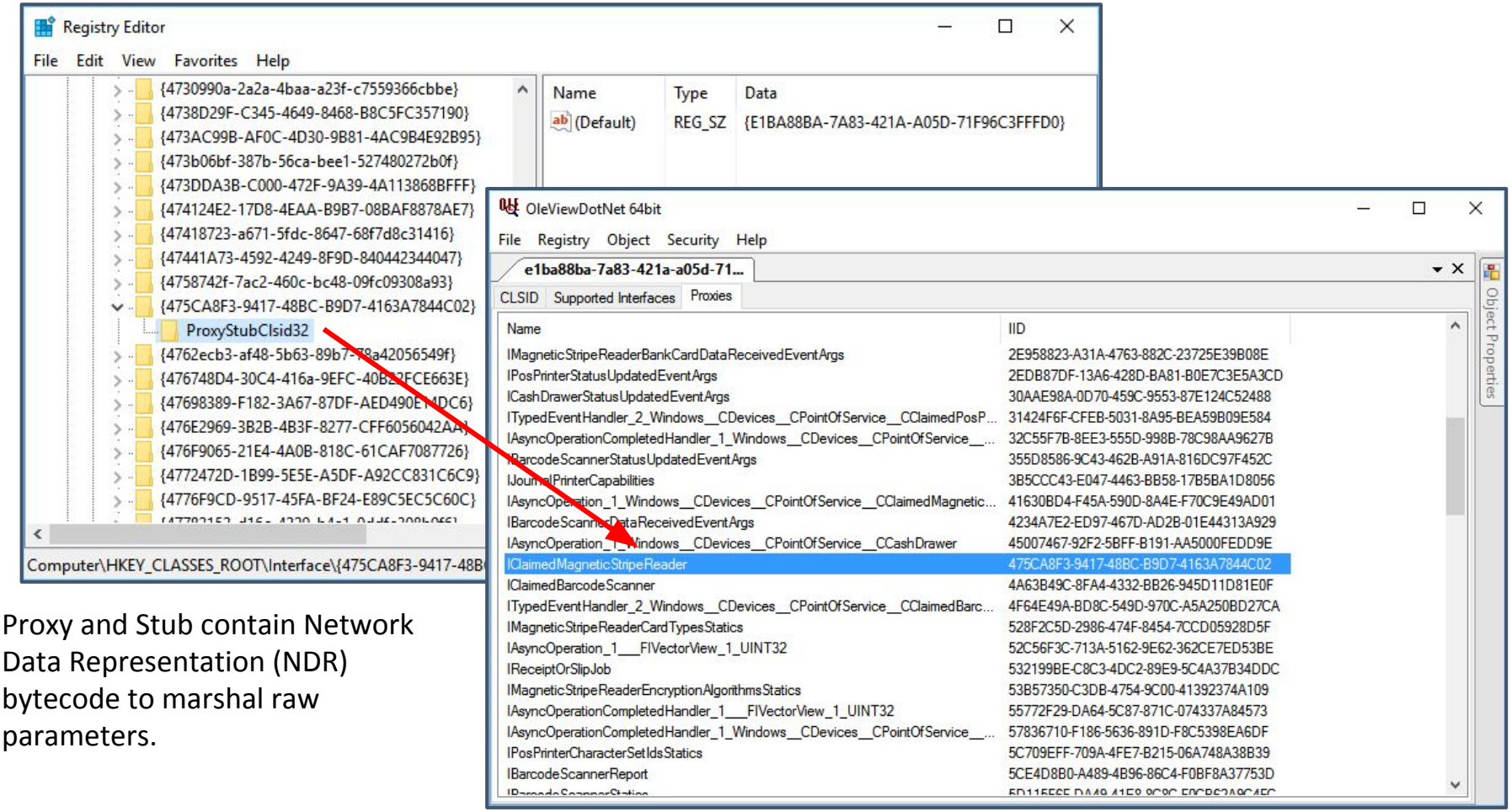
Standard Unmarshaling



Standard Unmarshaling



Interface Proxies and Stubs



Proxy and Stub contain Network Data Representation (NDR) bytecode to marshal raw parameters.

Proxy-Stub Factory

- Proxy-Stub CLSID doesn't use IClassFactory. Instead uses IPSFactoryBuffer.

```
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

- Proxies don't directly forward QueryInterface/AddRef/Release to the remote object, these calls are used on the proxy object.
- Actual IUnknown calls done on special interface, IRemUnknown

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

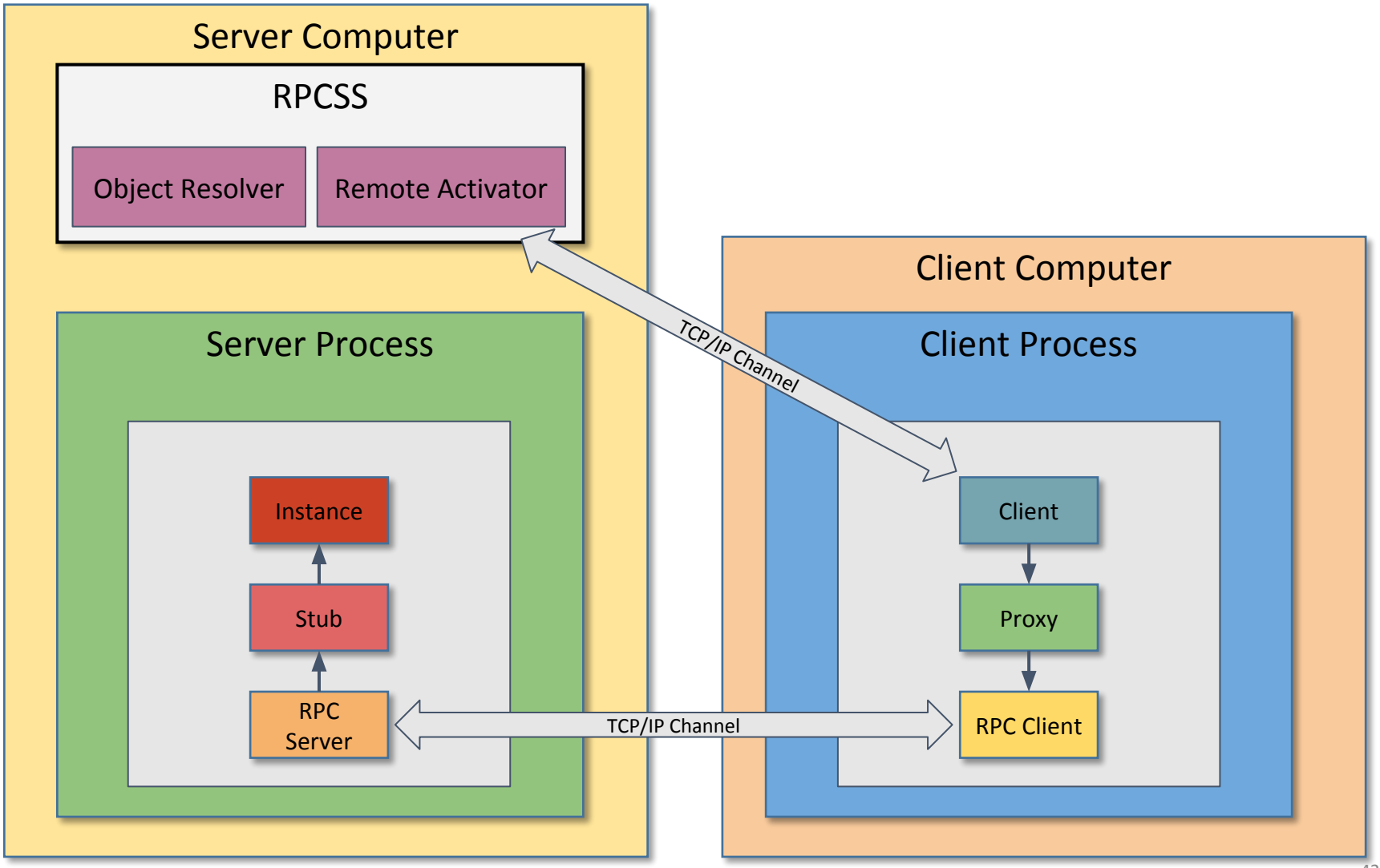
Returned in ActivationPropertiesOut

```
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



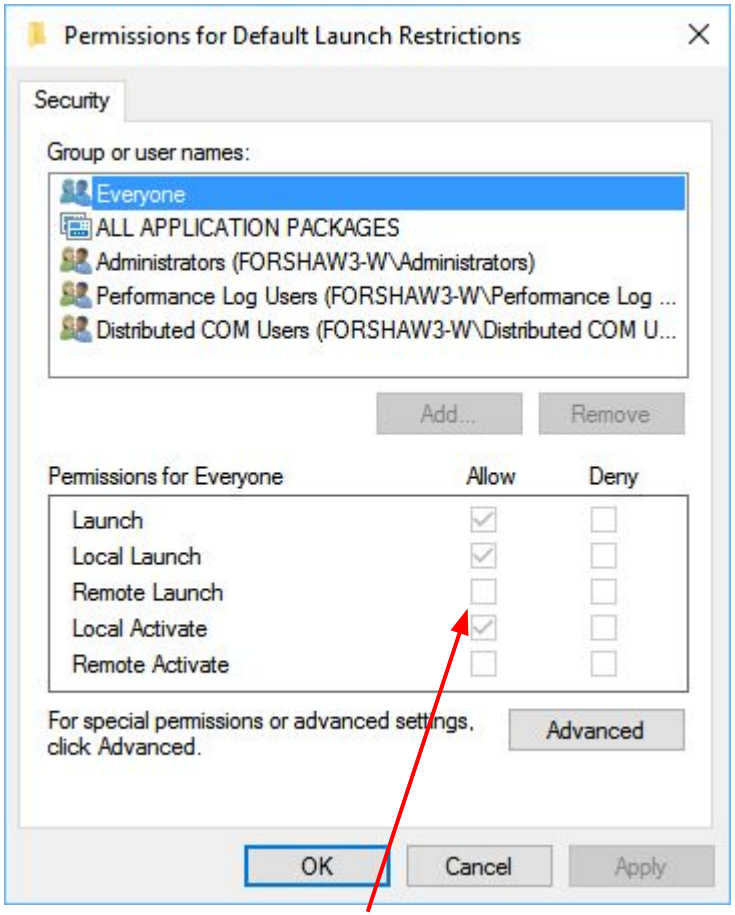
COSERVERINFO

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

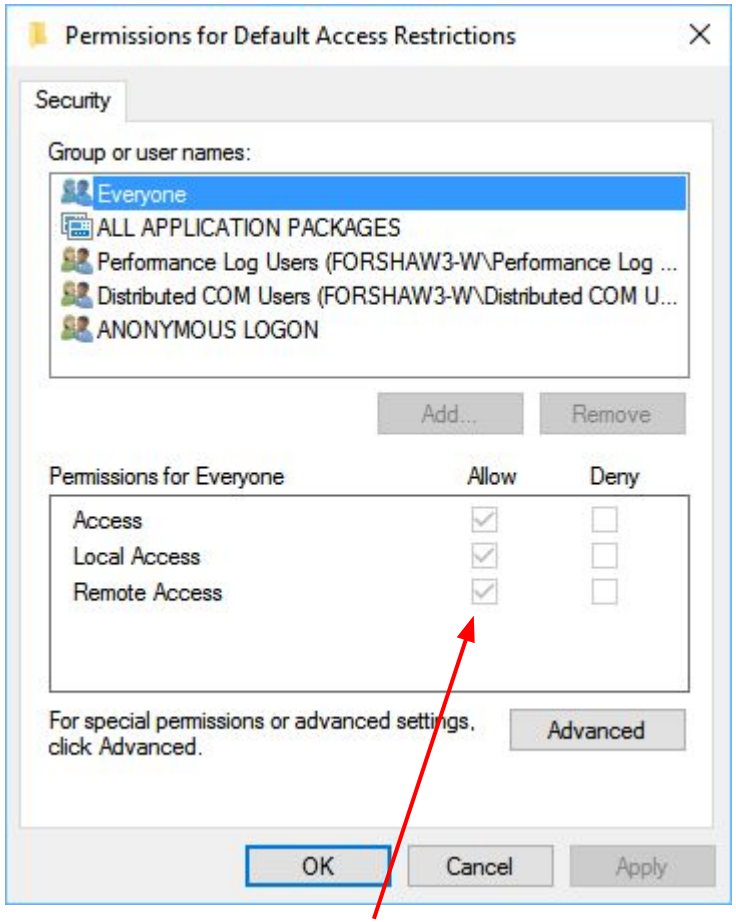
```
struct COAUTHINFO {
  DWORD dwAuthnSvc;
  DWORD dwAuthzSvc;
  LPWSTR pwszServerPrincName;
  DWORD dwAuthnLevel;
  DWORD dwImpersonationLevel;
  void* pAuthIdentityData;
  DWORD dwCapabilities;
};
```

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

COM Security Restrictions

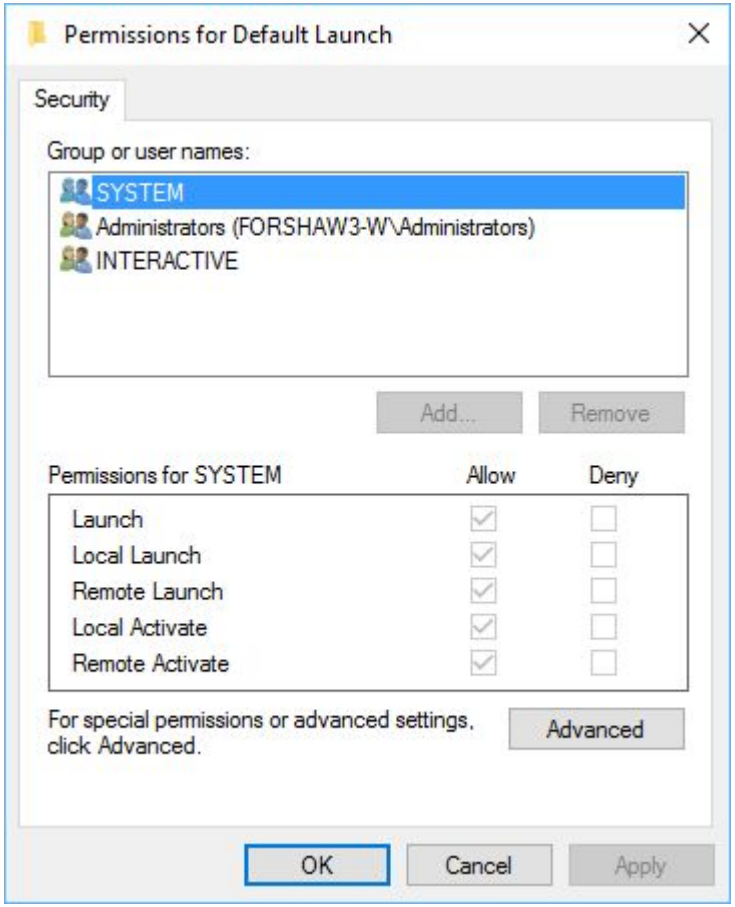


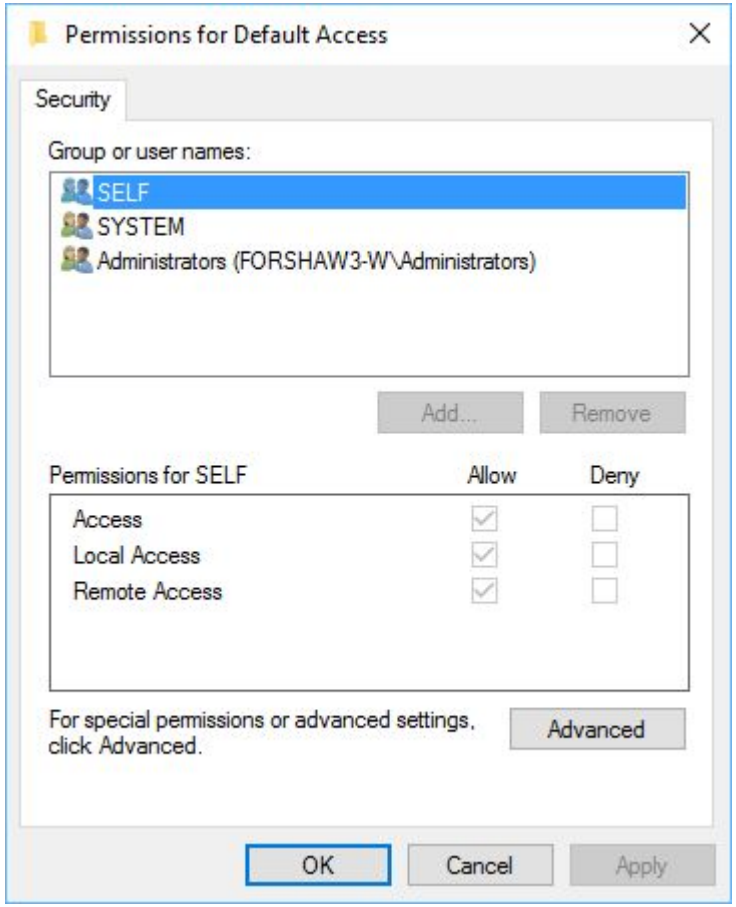
'Everyone' not allowed to launch or activate a new object remotely



'Everyone' can access an existing object remotely

COM Security



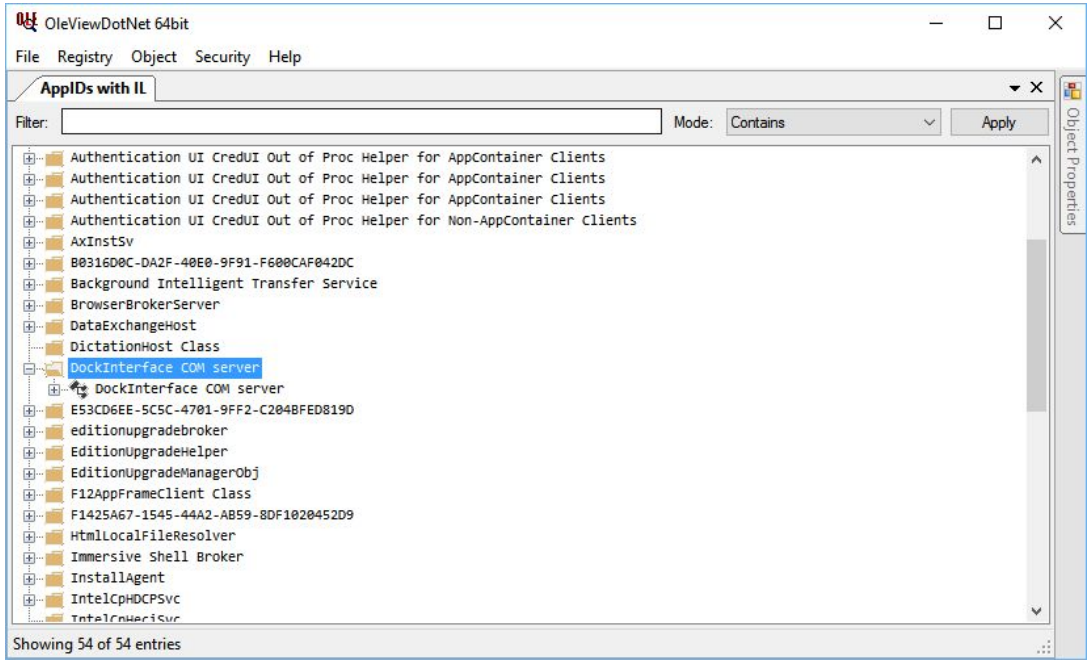


Launch = Create a new instance of the server.
Activate = Create new object on existing server.
Enforced in RPCSS

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

Integrity Levels

- Launching/Activating an OOP COM object from a Low IL needs an explicit entry in the SD for that IL.
- Accessing an object however DOES NOT need an IL label in the Access SD.



Security Through CoInitializeSecurity

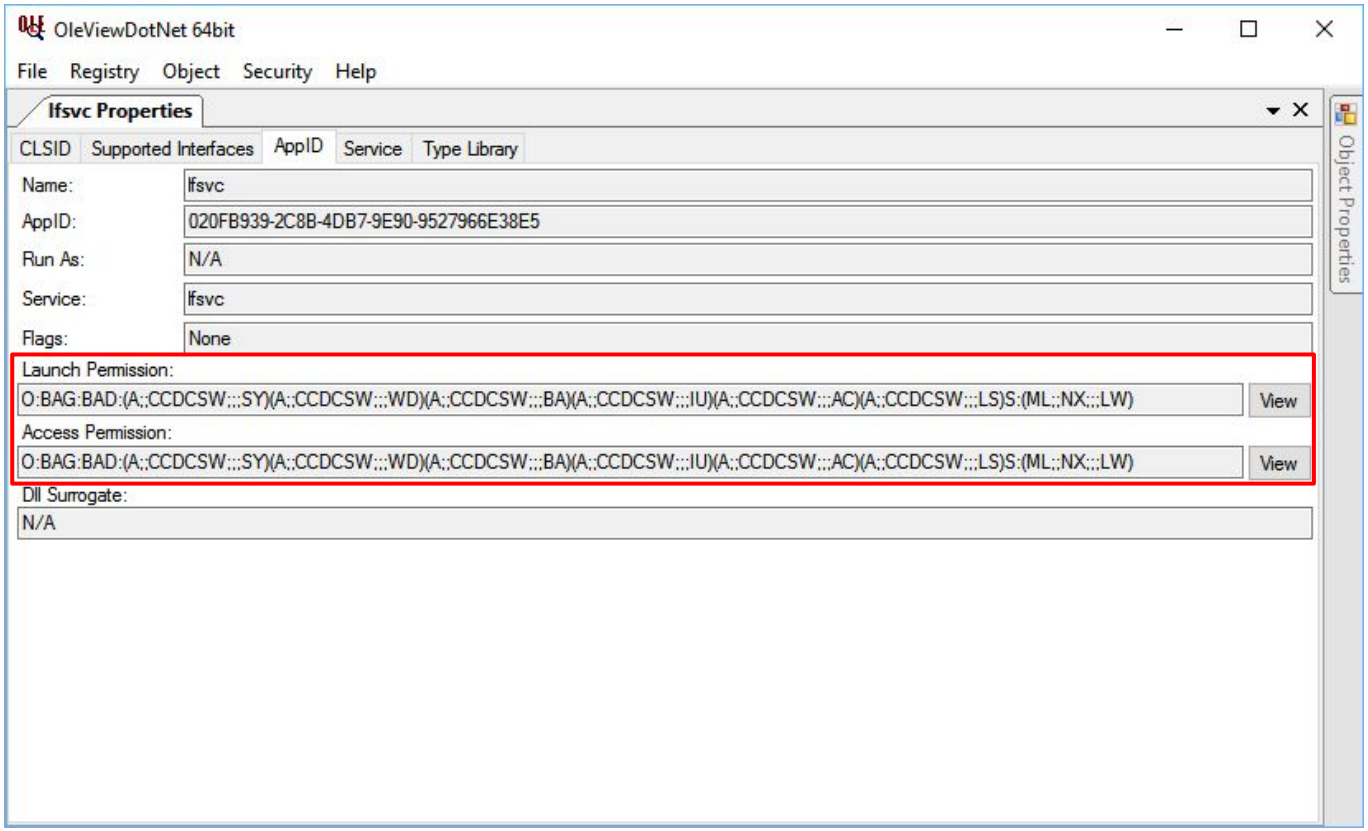
```
HRESULT CoInitializeSecurity(
    PSECURITY_DESCRIPTOR pSecDesc,
    LONG cAuthSvc,
    SOLE_AUTHENTICATION_SERVICE * asAuthSvc,
    void * pReserved1,
    DWORD dwAuthnLevel,
    DWORD dwImpLevel,
    void * pAuthList,
    DWORD dwCapabilities,
    void * pReserved3
);
```

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.

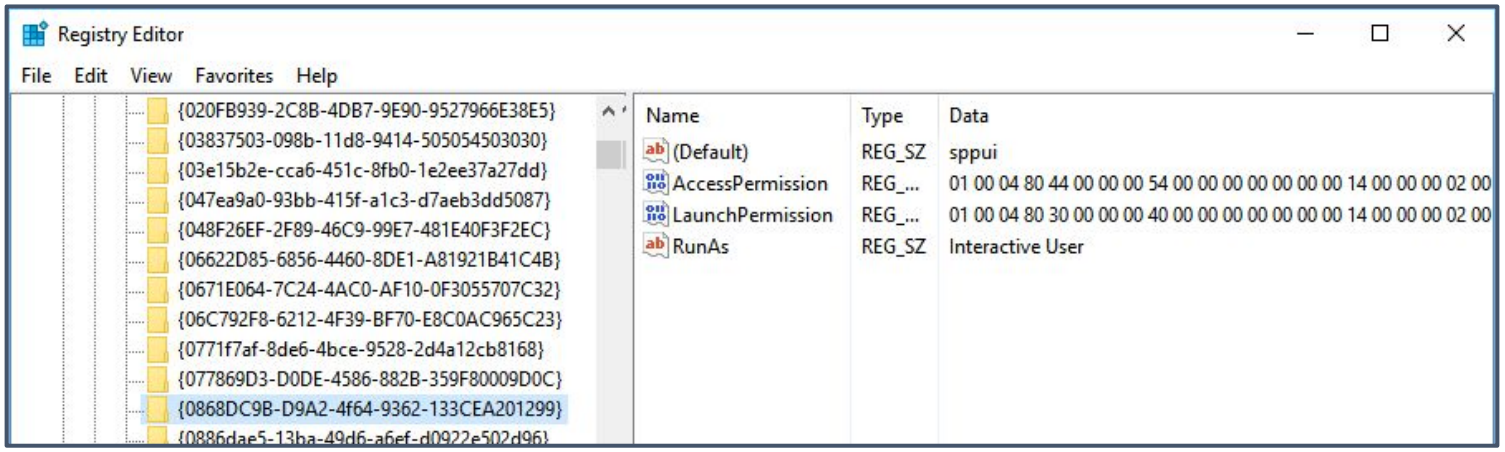
Security Through AppID

- Launch/Activation and Access Security can be specified through the AppID registration for the Class

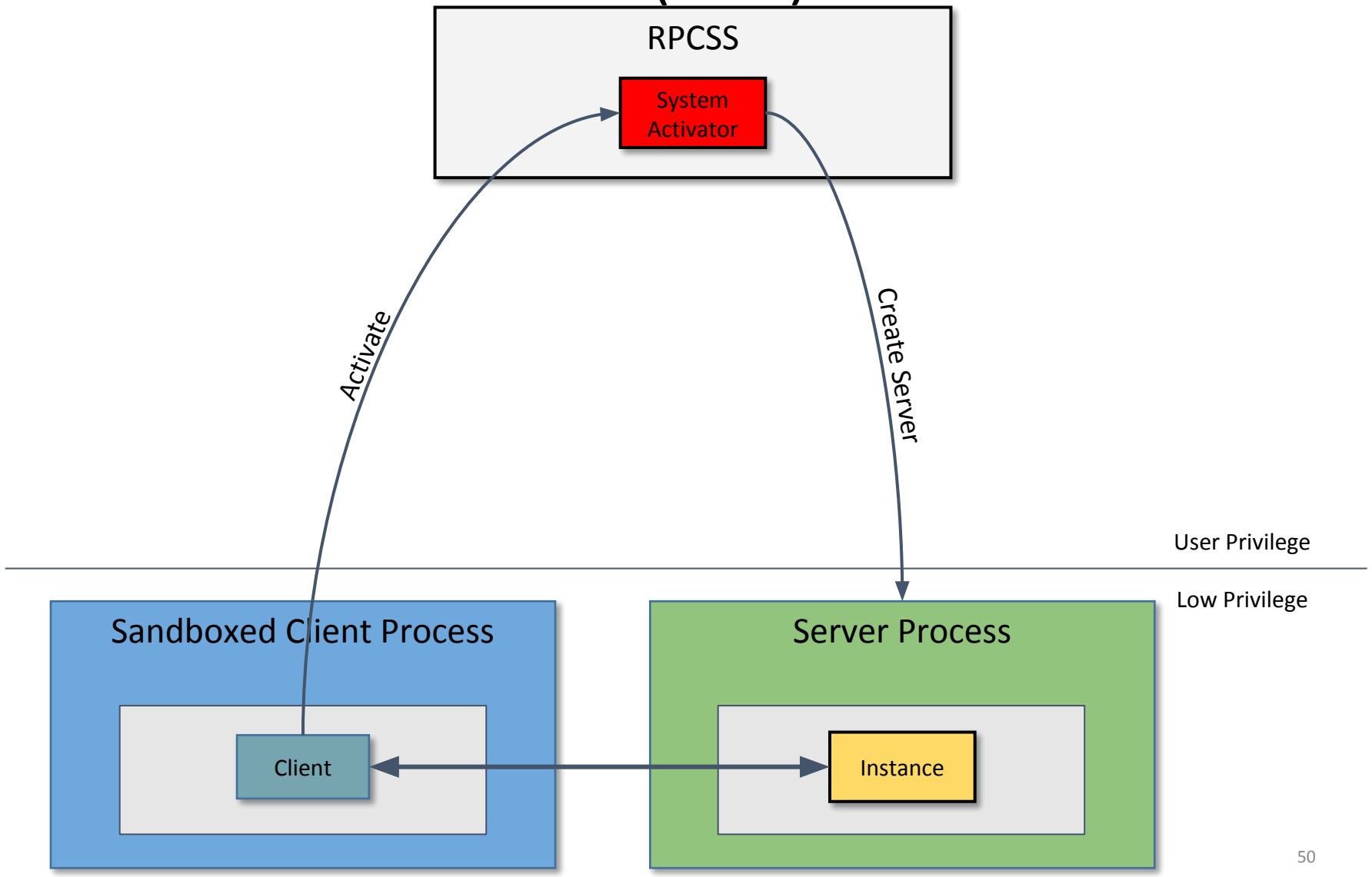


Application IDs

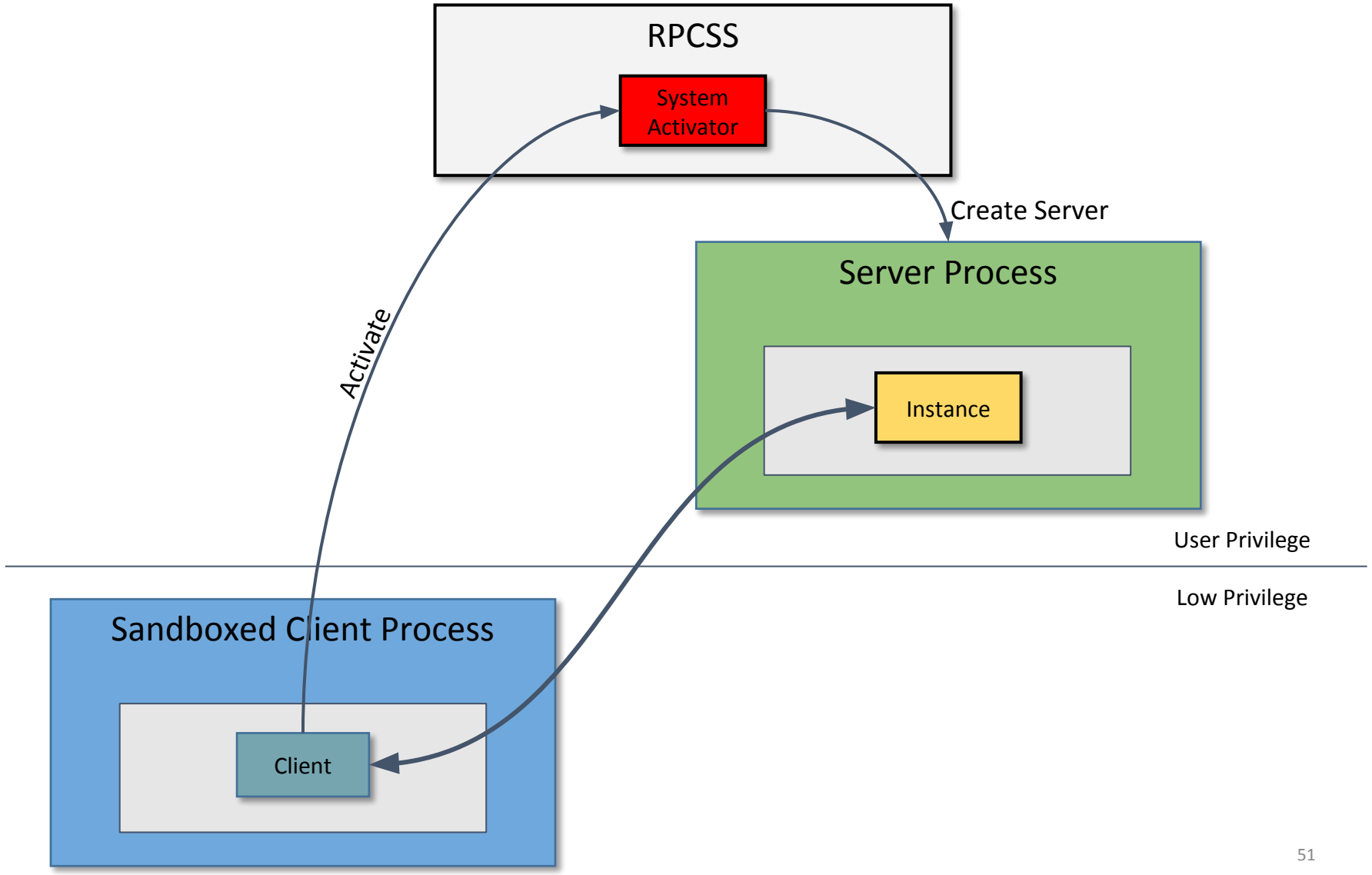
- AppIDs can configure more than just Launch/Access security
- Used to specify a number of features:
  - Allows you to specify a DLL Surrogate. This allows you to create in-process DLL servers as OOP local servers
  - Specify the object is hosted in a Windows service rather than a separate executable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



Activate as Activator (AAA)



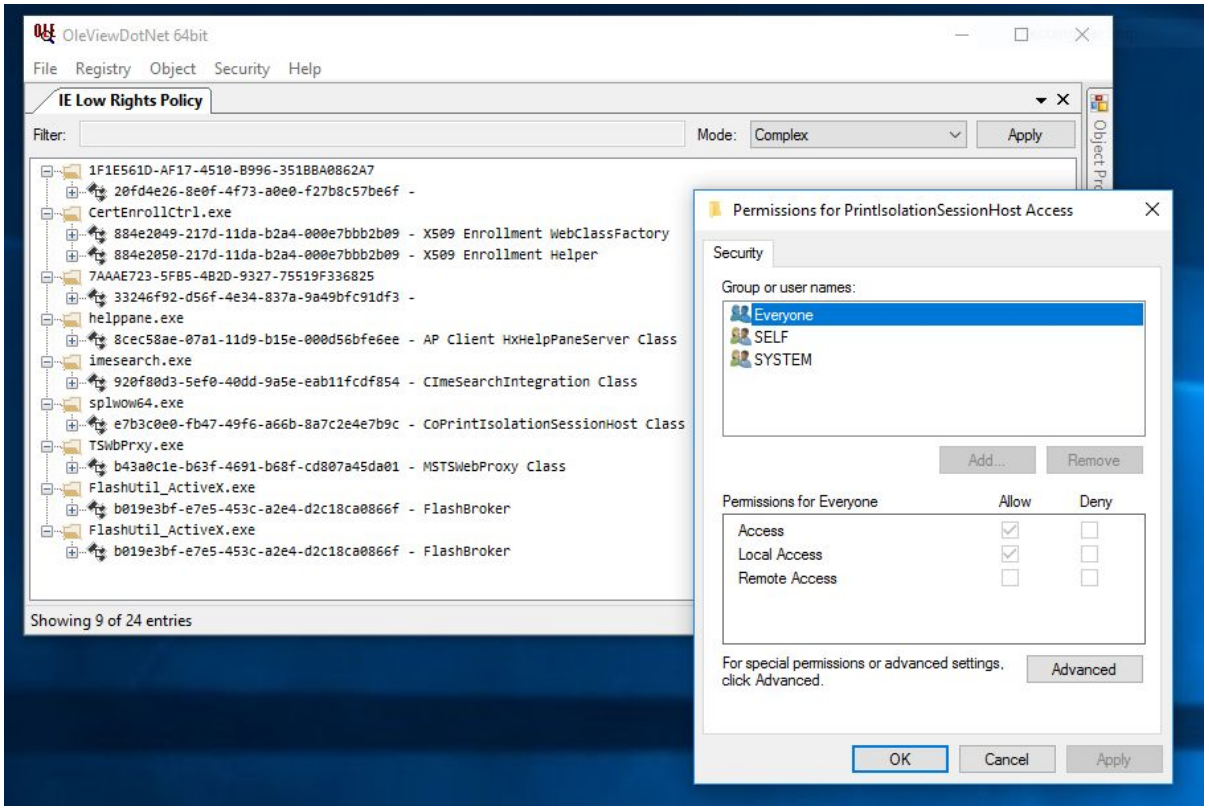
RunAs Interactive User



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

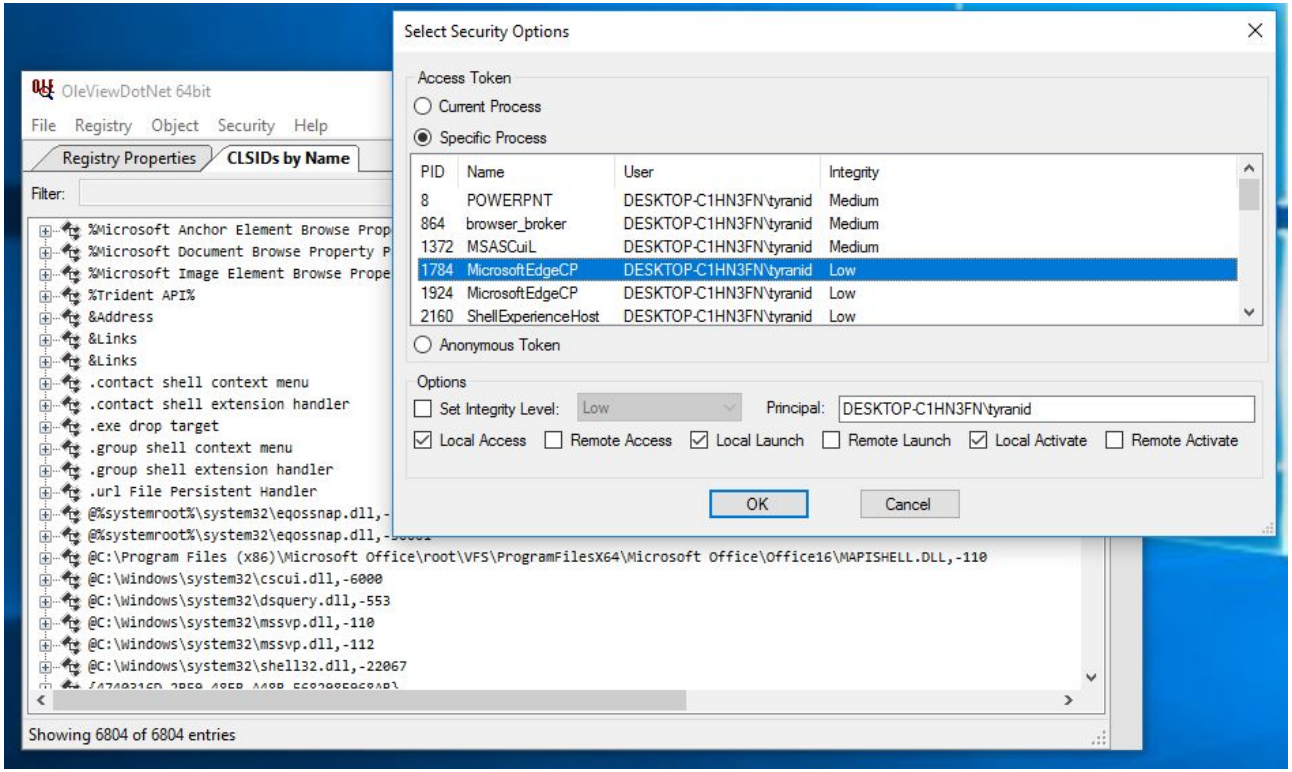
Accessible Objects from a Sandbox

- There's two ways to get a reference to a COM object
  - Activate via a broker (such as IE elevation policy)



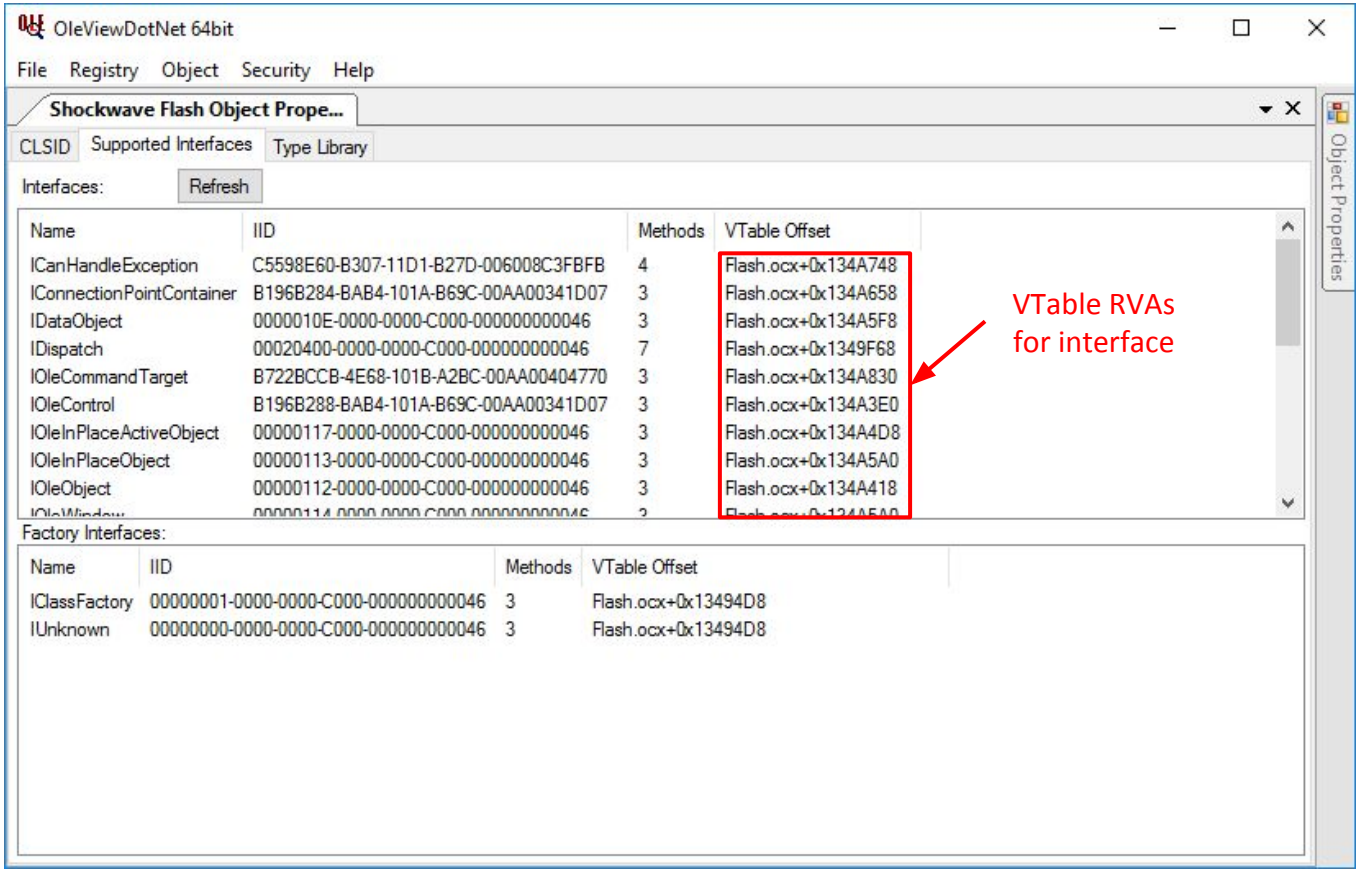
Accessible Objects from a Sandbox

- There's two ways to get a reference to a COM object
  - Activate via a broker (such as IE elevation policy)
  - Launch/Activate one directly



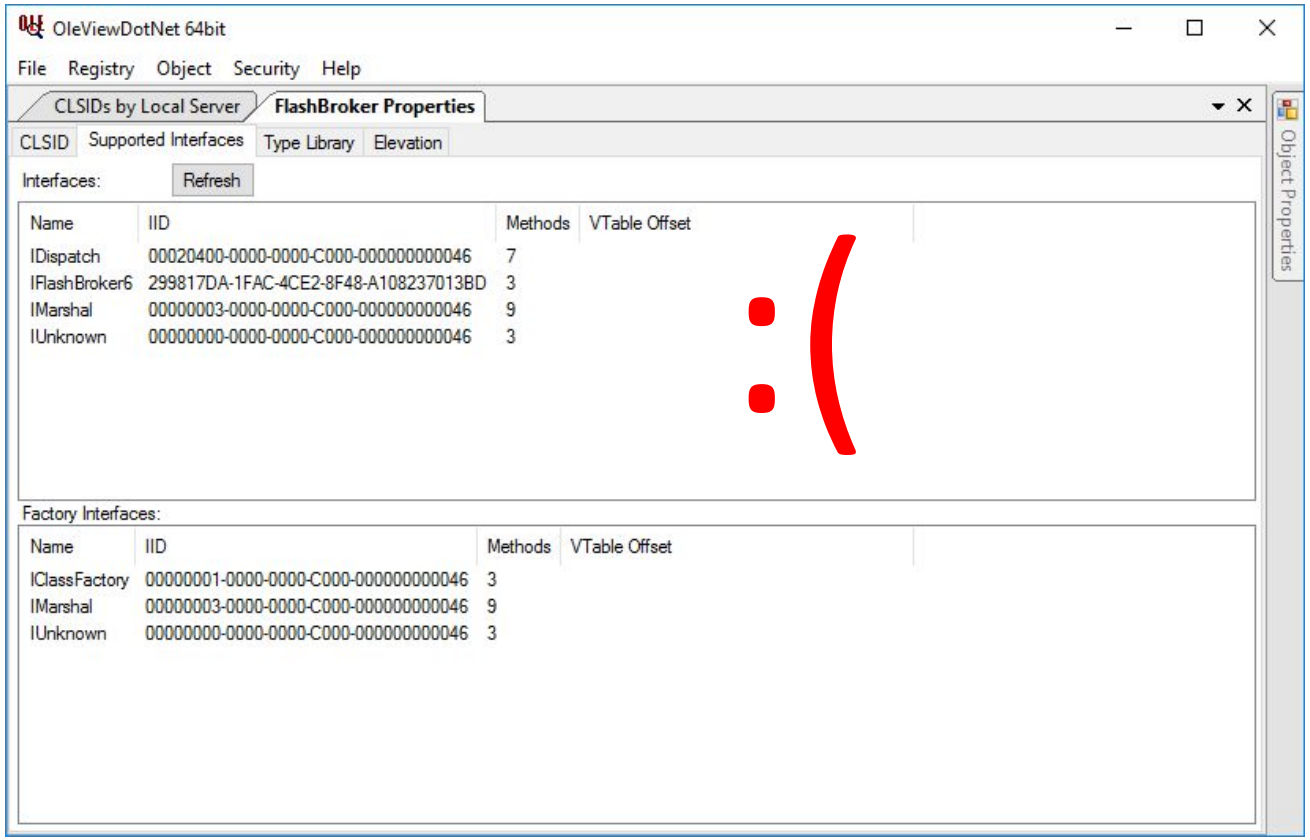
Reverse Engineering COM Components

- In-process you can just inspect the vtable address.



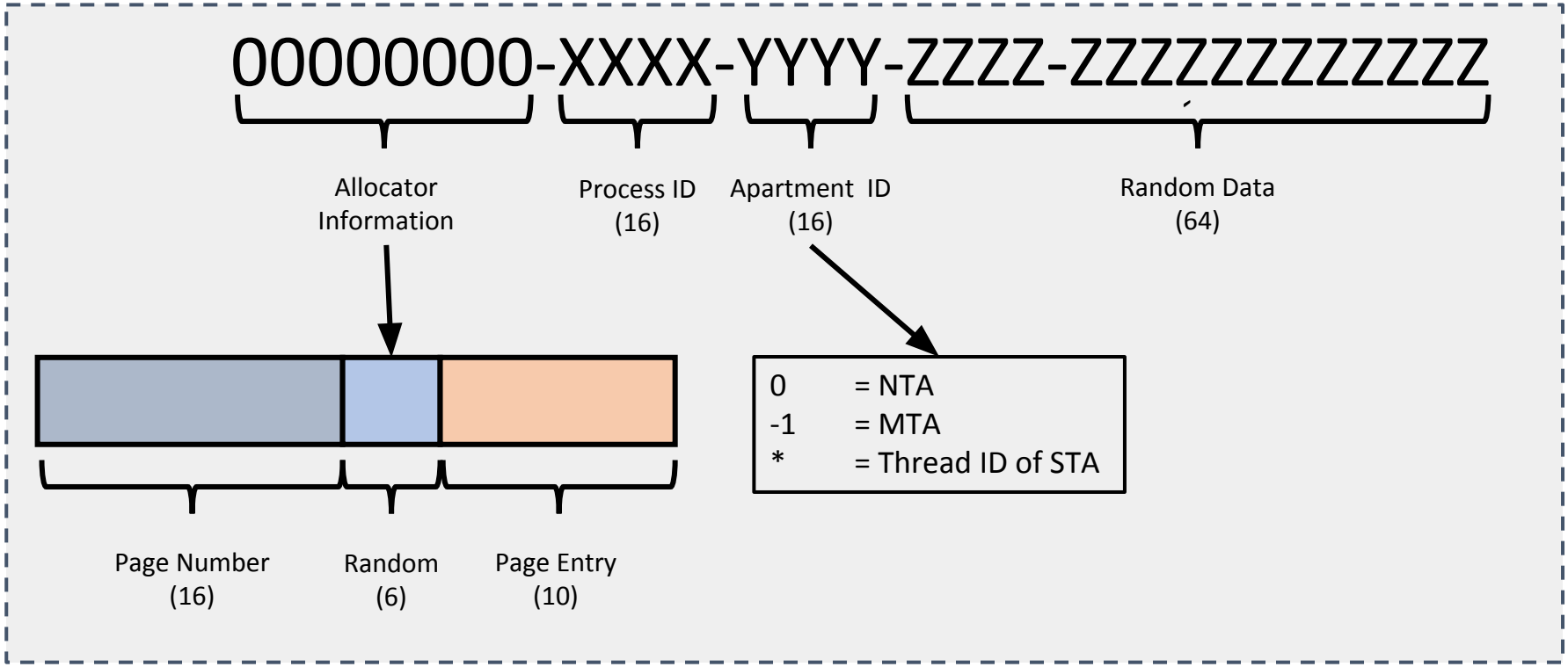
Reverse Engineering COM Components

- No such luck for OOP objects.

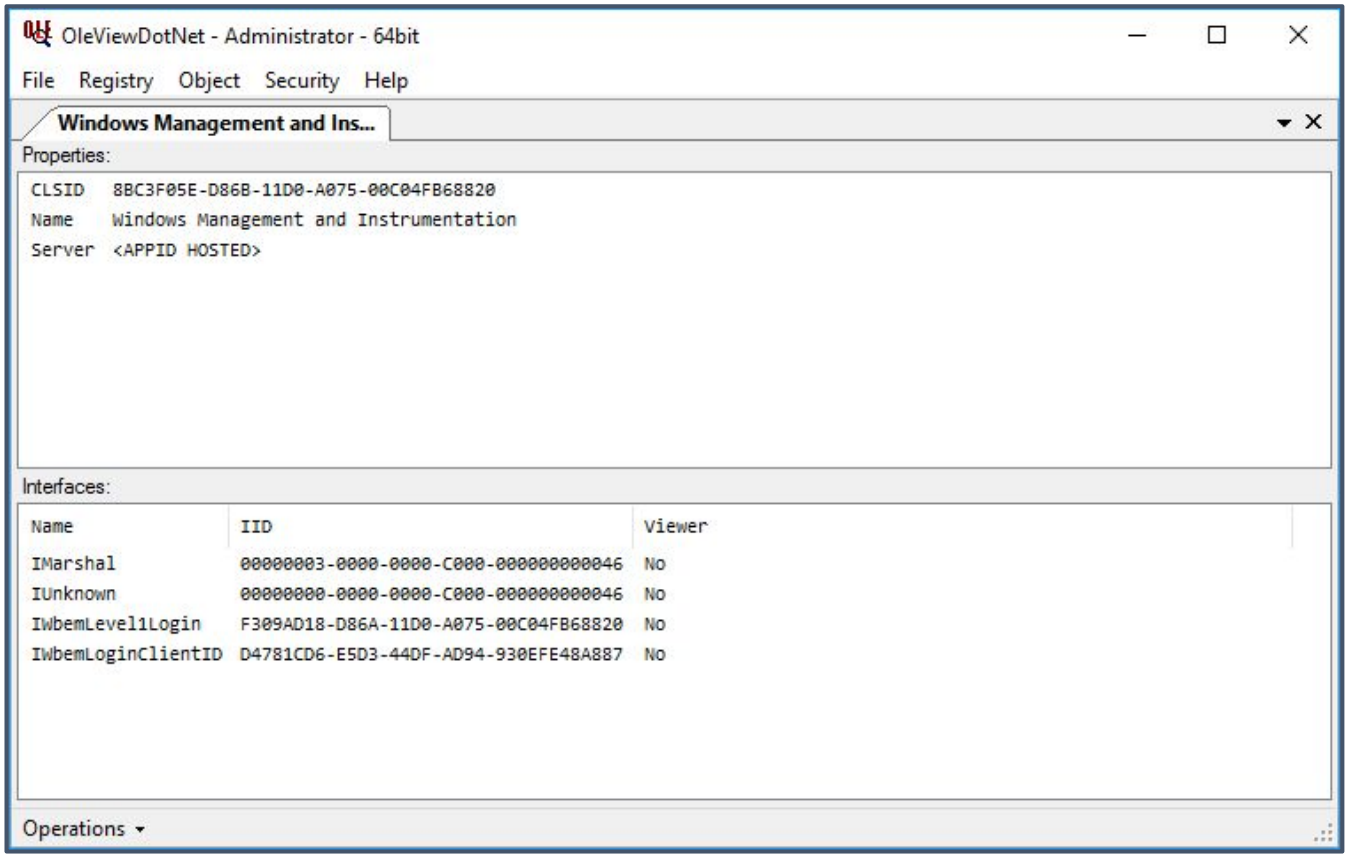


IPID Structure

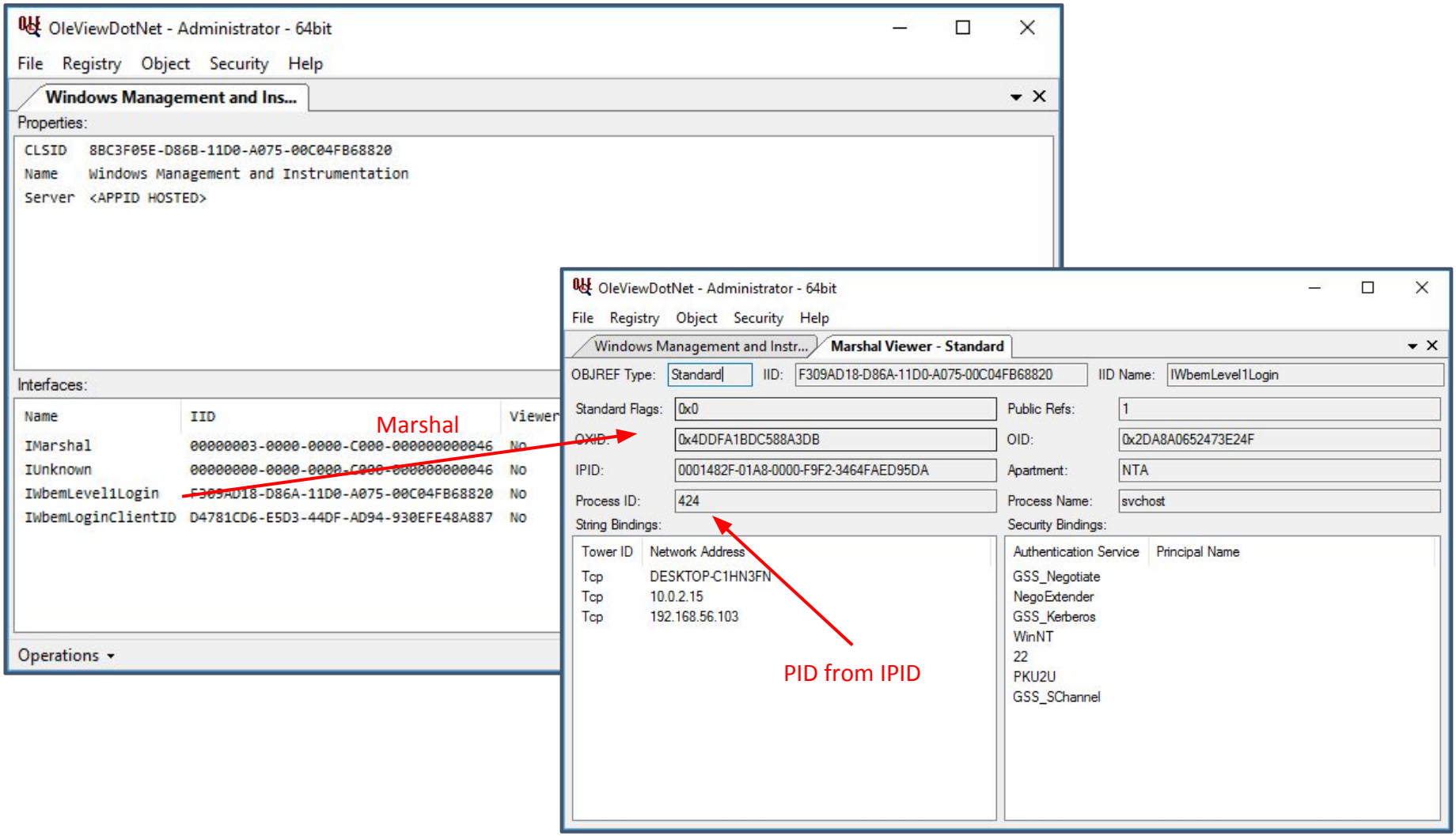
- You'd assume that the IPID was a random GUID, but nope.



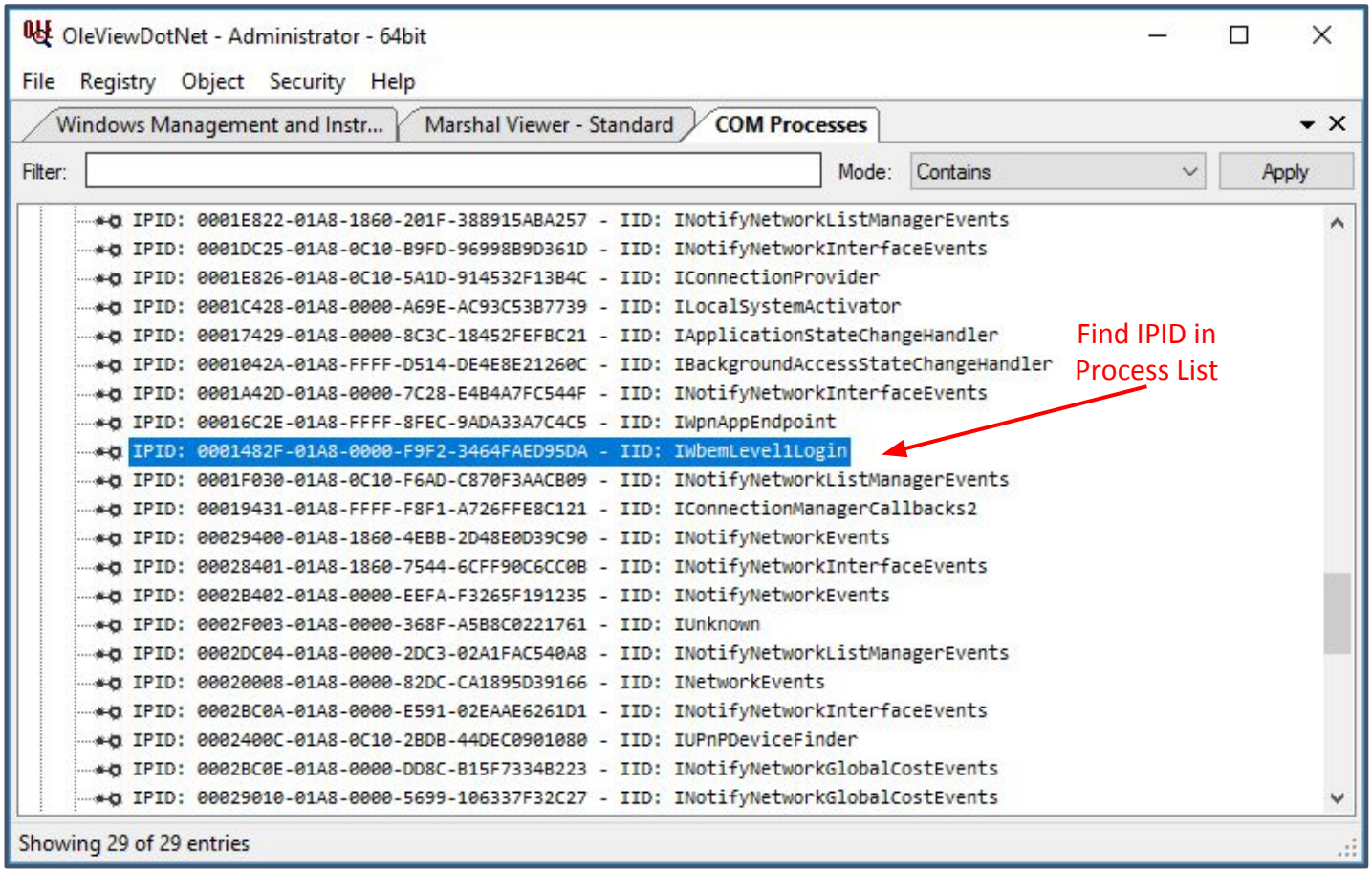
Tracking Down OOP VTable



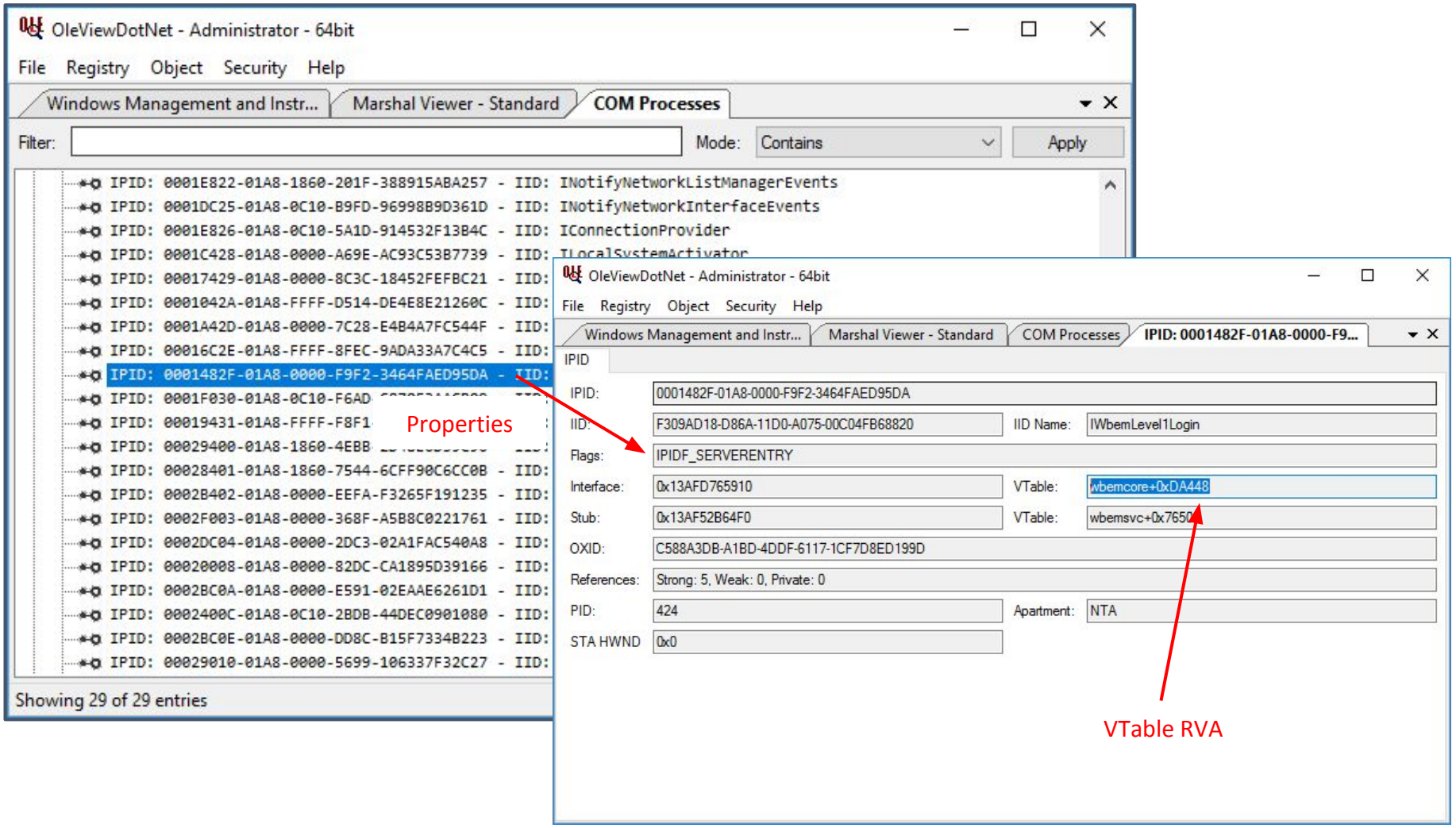
Tracking Down OOP VTable



Tracking Down OOP VTable



Tracking Down OOP VTable



VTable Reverse Engineering

```
.rdata:00000001800D79E0 ; const CJobManagerExternal::`vftable'
.rdata:00000001800D79E0 ??_7CJobManagerExternal@@6B@ dq offset ?QueryInterface@?$RuntimeClass@U?$InterfaceList@U
.rdata:00000001800D79E0                                         ; DATA XREF: CJobManagerExternal::CJobManagerExt
.rdata:00000001800D79E0                                         ; CJobManagerExternal::~CJobManagerExternal(void
.rdata:00000001800D79E0                                         ; Microsoft::WRL::Details::RuntimeClass<Microsof
.rdata:00000001800D79E8                 dq offset ?AddRef@?$RuntimeClass@U?$InterfaceList@UIBackgroundCopyFile@@
.rdata:00000001800D79F0                 dq offset ?Release@?$RuntimeClass@U?$InterfaceList@UIBackgroundCopyManag
.rdata:00000001800D79F8                 dq offset ?CreateJob@CJobManagerExternal@@UEAAJPEBGW4__MIDL_IBackgroundC
.rdata:00000001800D7A00                 dq offset ?GetJob@CJobManagerExternal@@UEAAJAEBU_GUID@@PEAPEAUIBackgroun
.rdata:00000001800D7A08                 dq offset ?EnumJobs@CJobManagerExternal@@UEAAJKPEAPEAUIEnumBackgroundCop
.rdata:00000001800D7A10                 dq offset ?GetErrorDescription@CJobManagerExternal@@UEAAJJKPEAPEAG@Z ; C
.rdata:00000001800D7A18                 dq offset ??_ECJobManagerExternal@@UEAAPEAXI@Z ; CJobManagerExternal::`v
```

IUnknown

Interface Specific

QueryInterface Implementations

```
HRESULT QueryInterface(REFIID riid, void** ppv) {
  if (riid == IID_IUnknown || riid == IID_Interface1)
{
    *ppv = this;
  } else if (riid == IID_Interface2) {
    *ppv = static_cast<Interface2*>(this);
  } else {
    return E_NOINTERFACE;
  }
  ((IUnknown*)*ppv)->AddRef();
  return S_OK;
}
```

Explicit Implementation

```
HRESULT QueryInterface(REFIID riid, void** ppv) {
  QITAB qit[] = {{ &IID_Interface1, 0 },
    { &IID_Interface2, 8 } { NULL, 0 }};
  return QISearch(this, qit, riid, ppv);
}
```

Using QISearch Helper

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

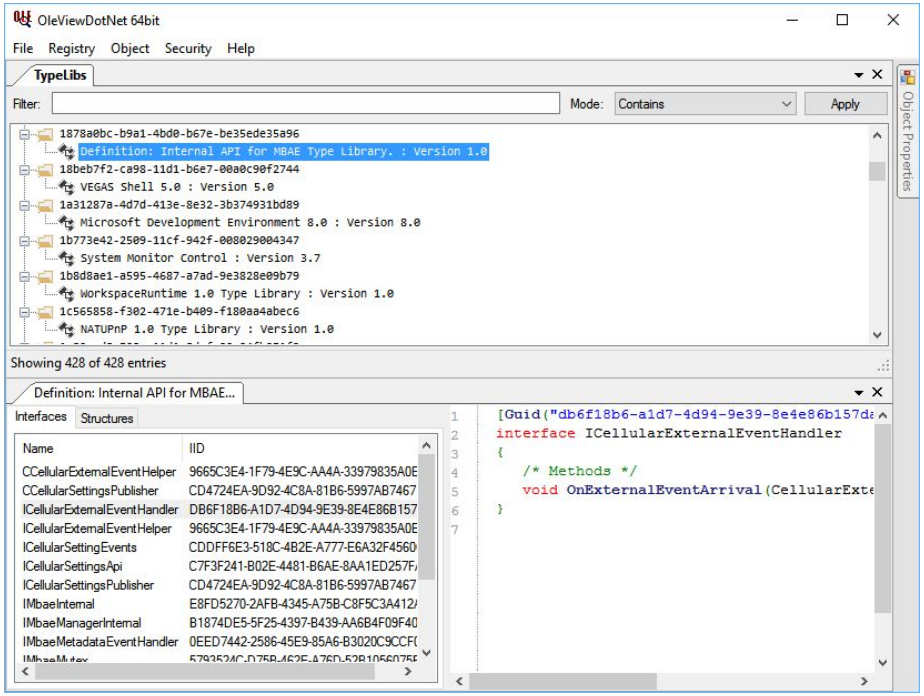
```
; __int32 __fastcall CJobManagerExternal::CreateJob(CJobManagerExternal *this, const unsigned __int16 *, enum __MIDL_IBackgroundCopyJob_0003, struct _GUID *, struct IBackgroundCopyJob **)
?CreateJob@CJobManagerExternal@@UEAAJPEBGW4__MIDL_IBackgroundCopyJob_0003@@PEAU_GUID@@PEAPEAUIBackgroundCopyJob@@@Z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

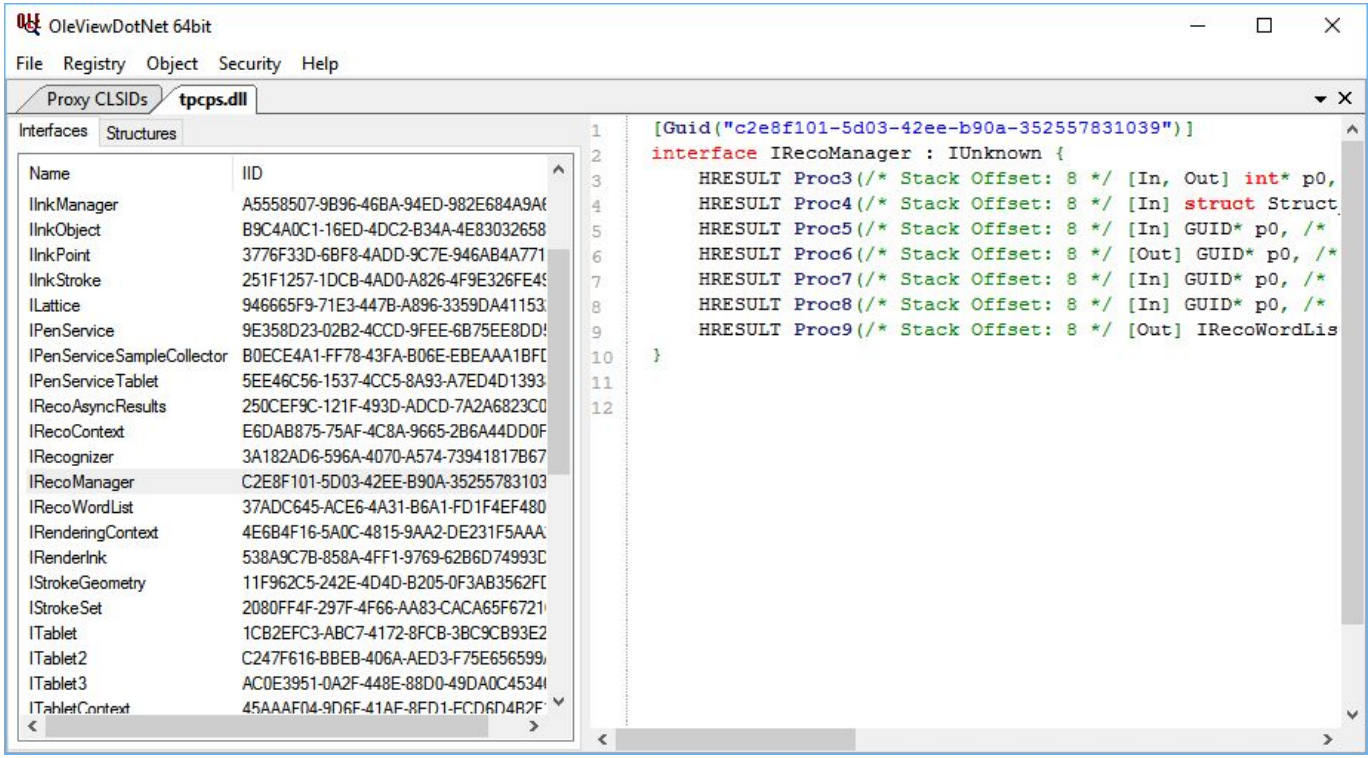
Interface Information - Type Libraries

- Some objects have type library information which is a structured format for describing COM interfaces.
- “import” the library into a C++ project with #import <name.tlb> preprocessor directory



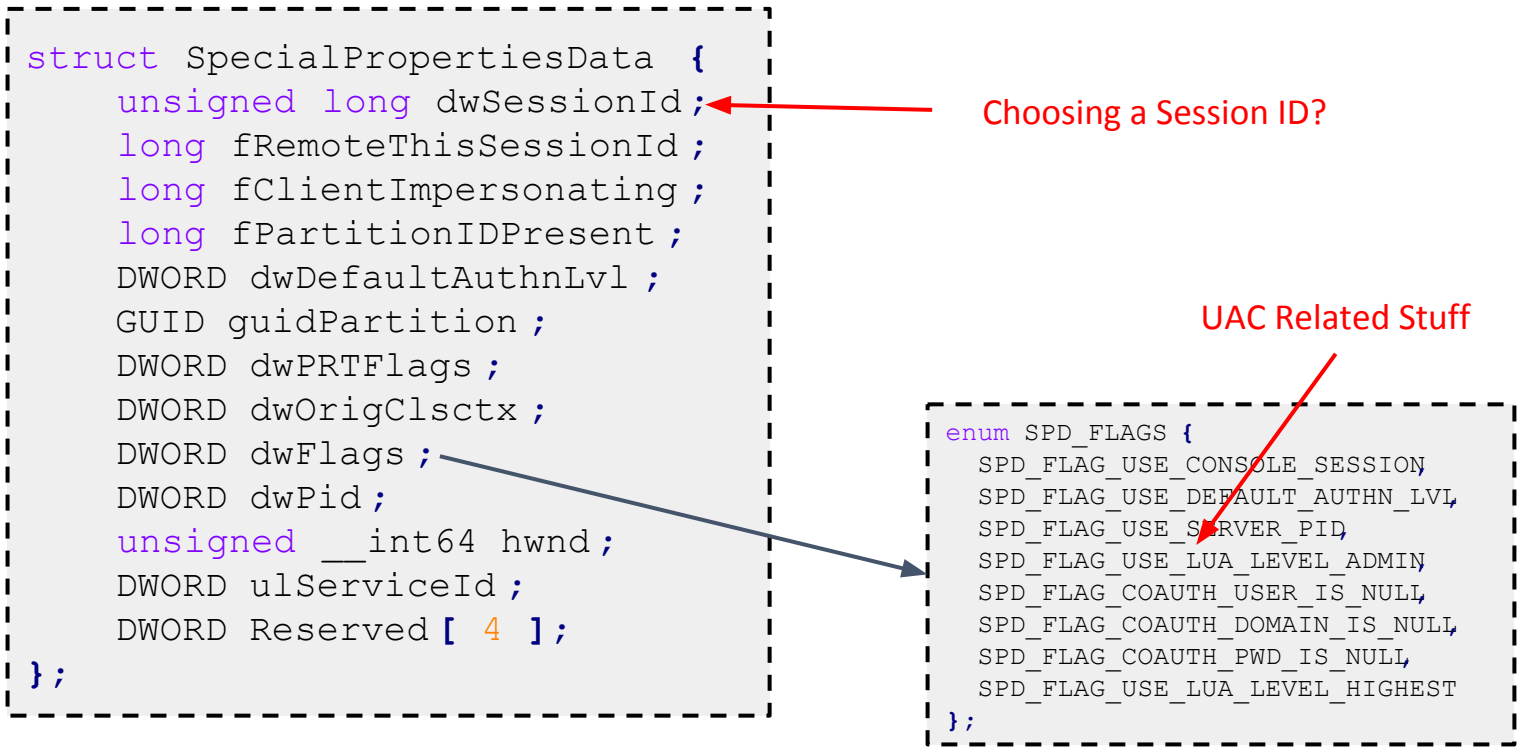
Interface Information - Proxy/Stub NDR

- Proxies and Stubs contains NDR bytecode which describe how to marshal parameters back and forth.
- Contains information about interfaces and structures



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

Activation Properties In- SPD



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.

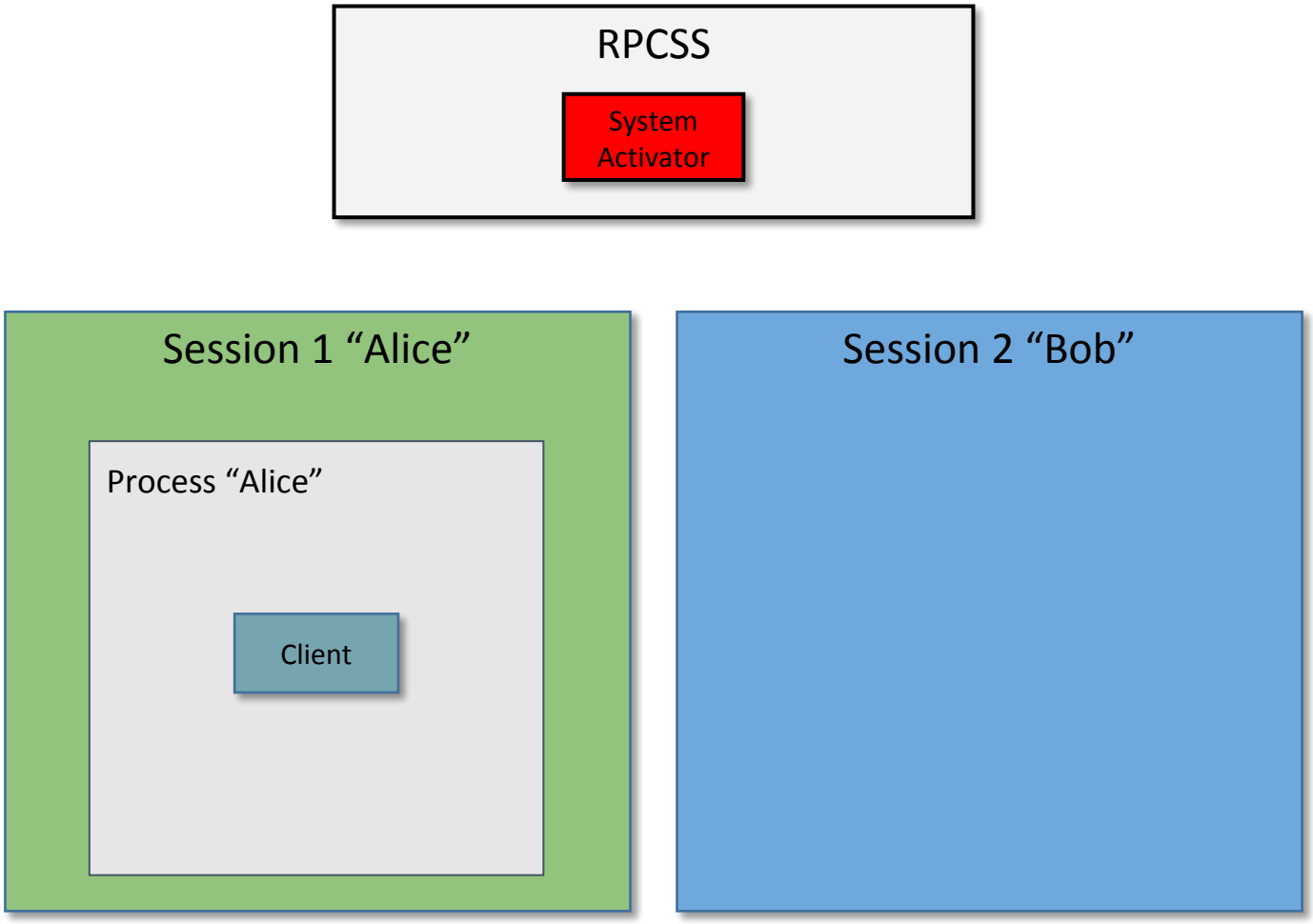
Monikers and Binding

- Monikers are string representation of COM objects.
- Pass string to CoGetObject to “Bind” to the underlying object.

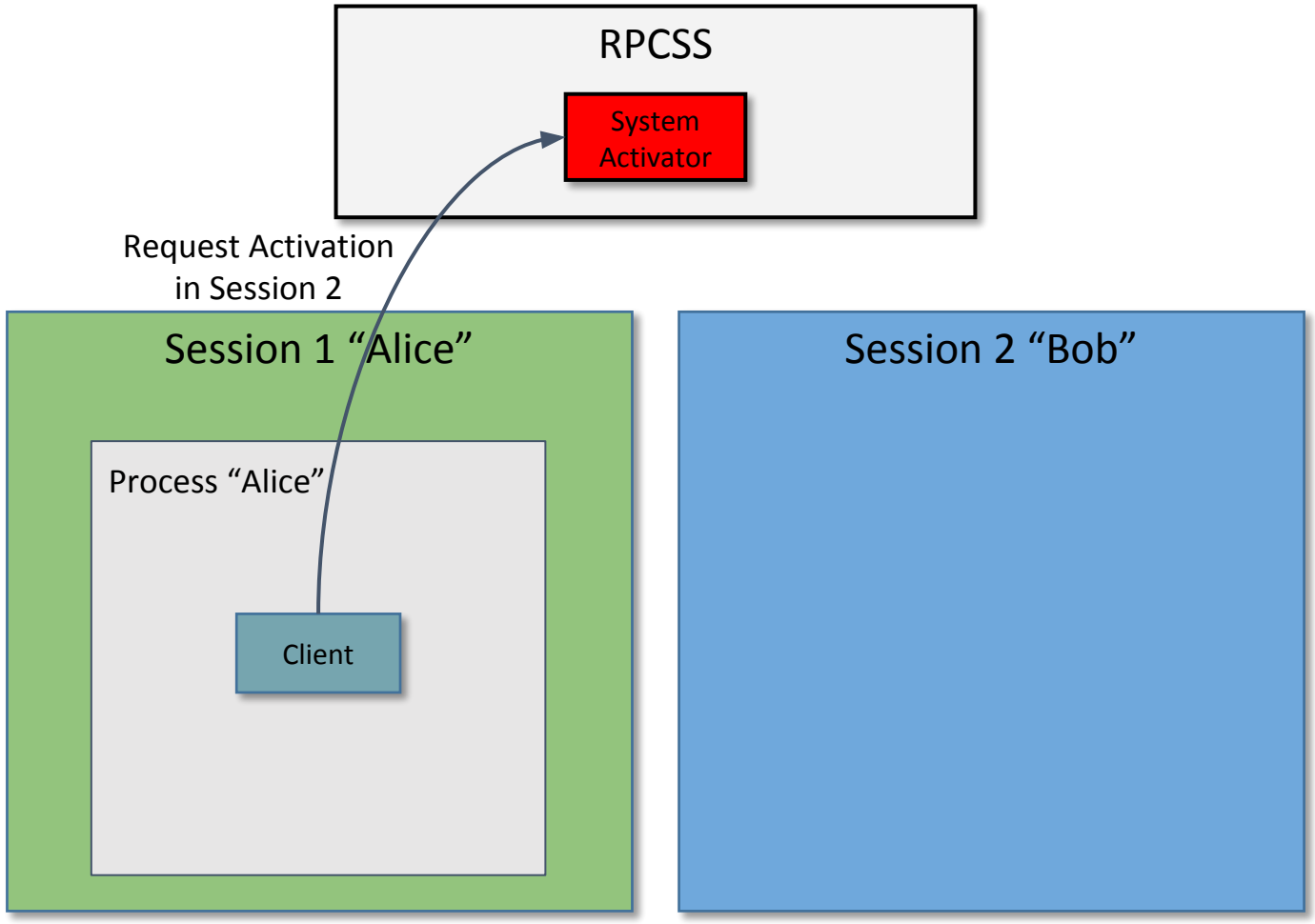
```
HRESULT CoCreateInstanceInSession(DWORD session,
    REFCLSID rclsid, REFIID riid, void ** ppv) {
    BIND_OPTS3 bo = {};
    WCHAR  wszCLSID[50];
    WCHAR  wszMonikerName[300];

    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);
}
```

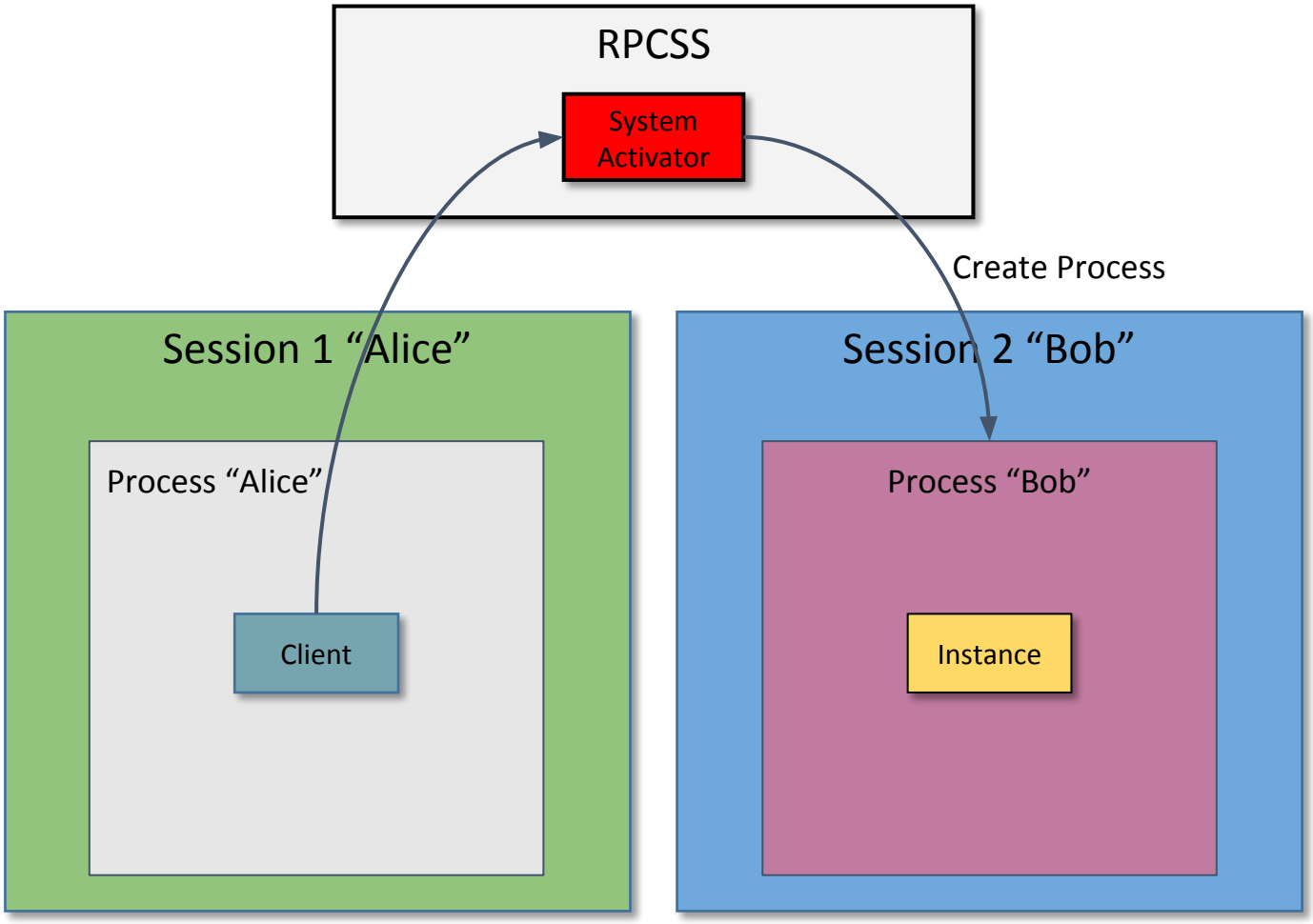
Session Moniker in Action



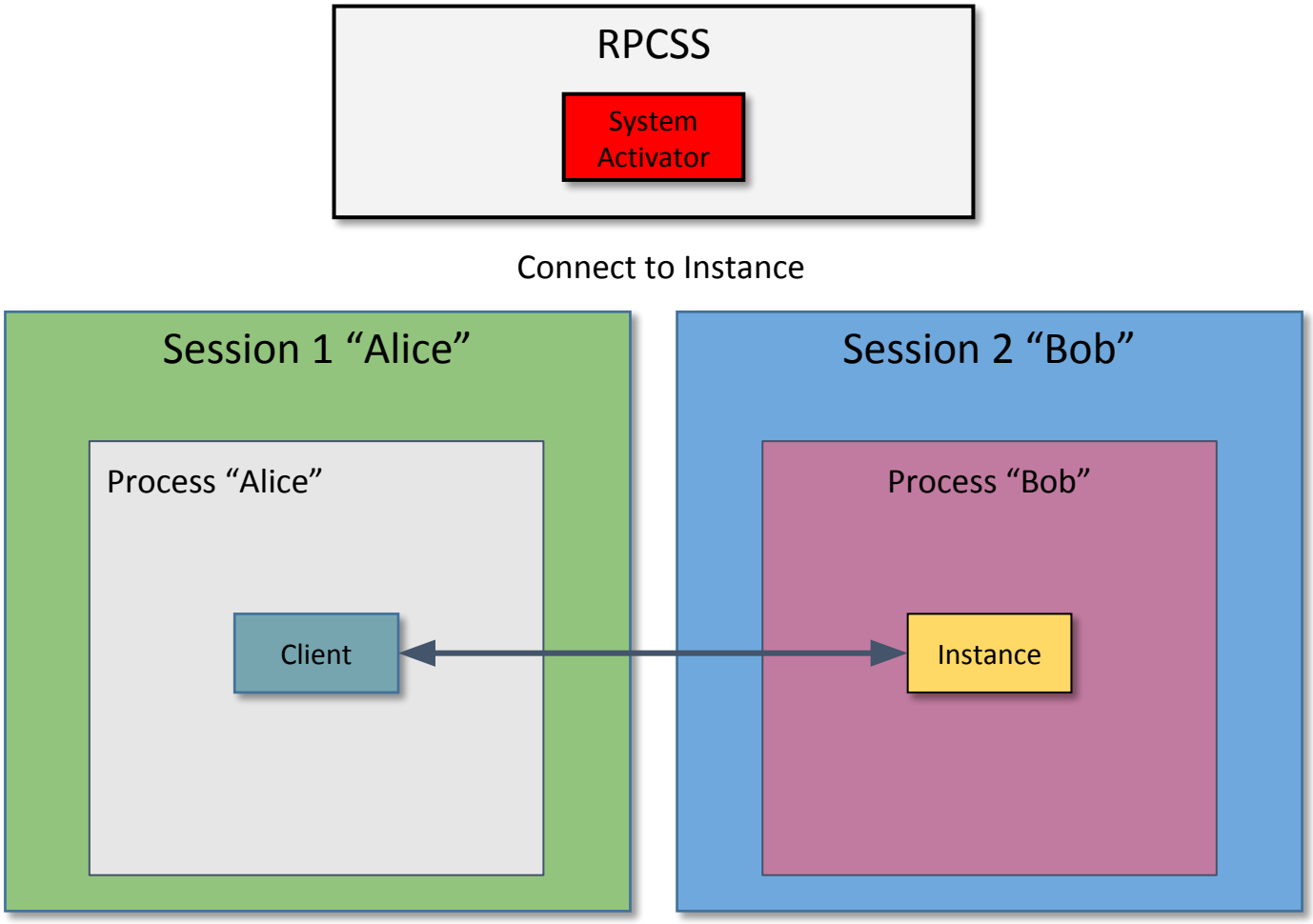
Session Moniker in Action



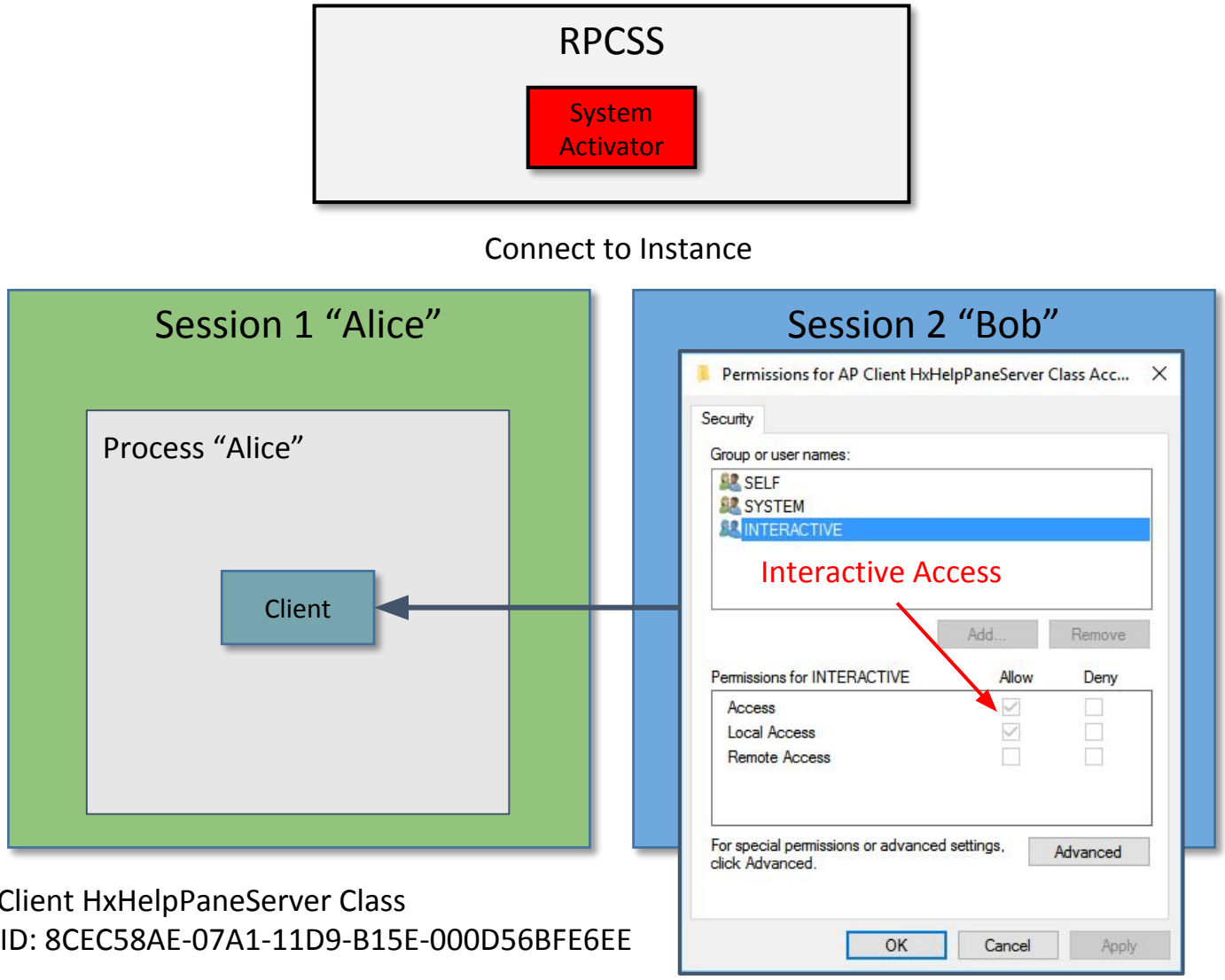
Session Moniker in Action



Session Moniker in Action

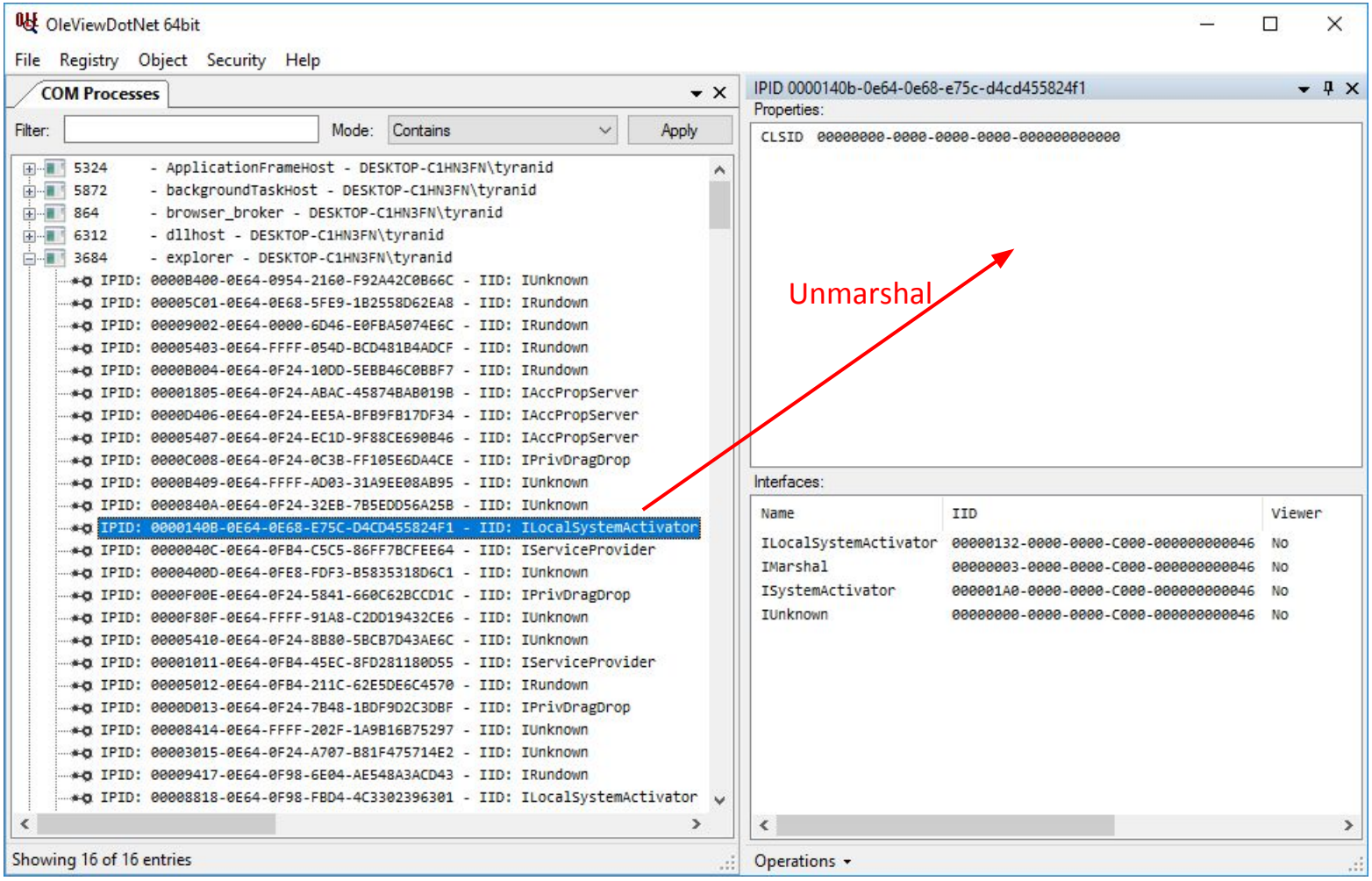


Session Moniker in Action



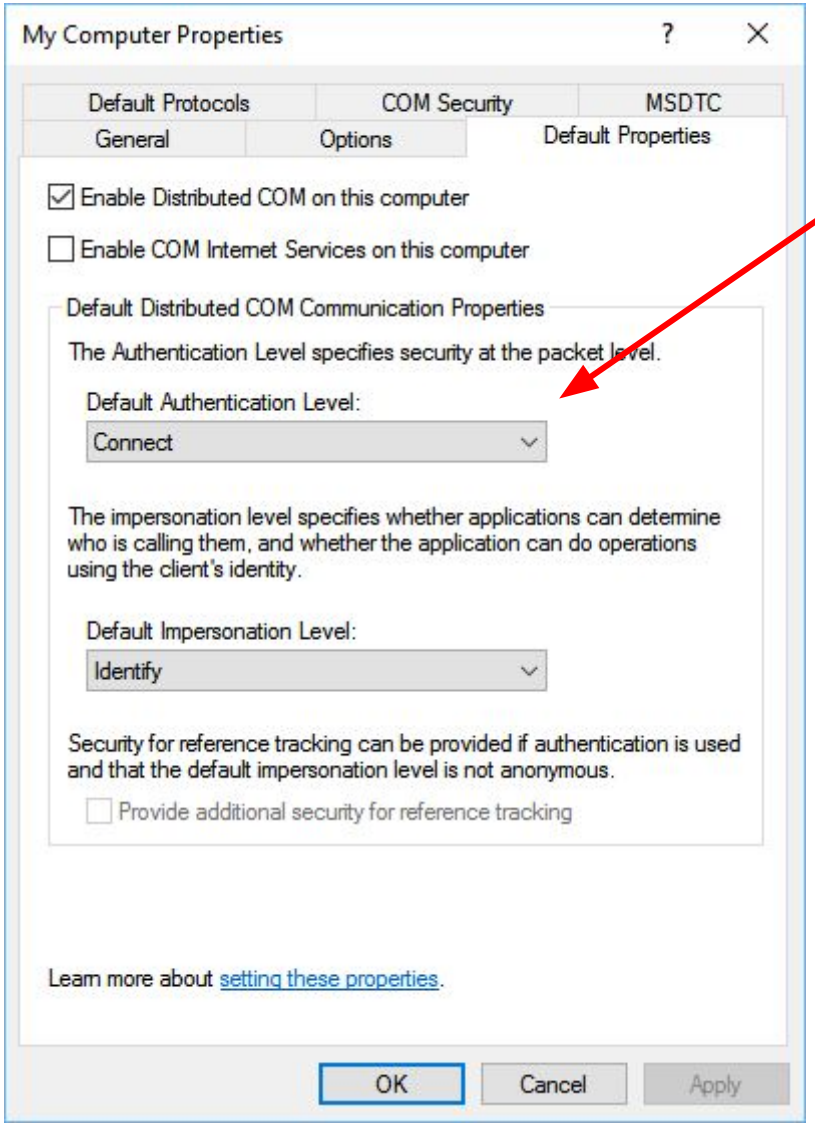
AP Client HxHelpPaneServer Class
CLSID: 8CEC58AE-07A1-11D9-B15E-000D56BFE6EE

Abusing Information Disclosure



You Can’t Help Loving DCOM

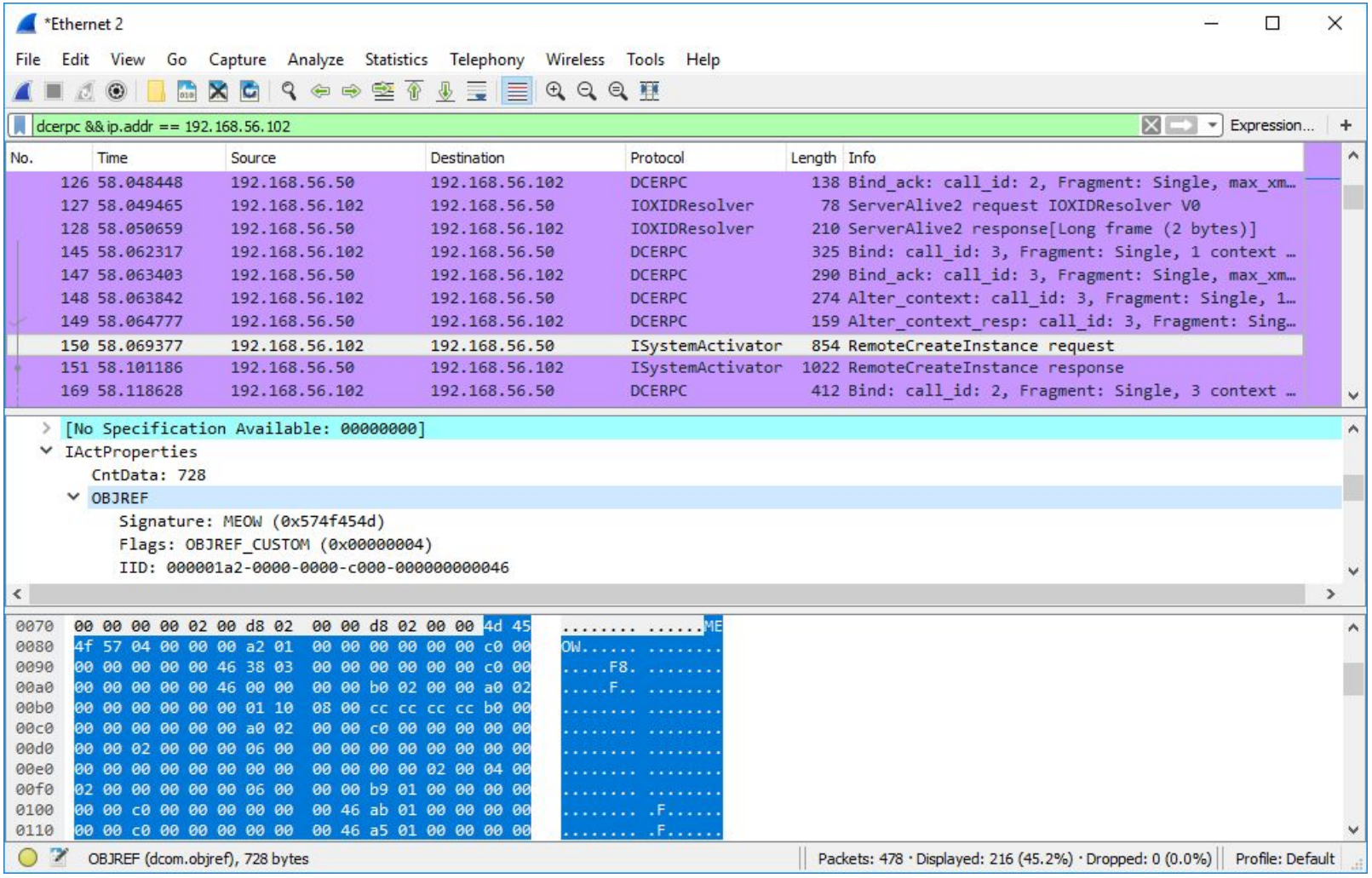
Plain Text Communications?



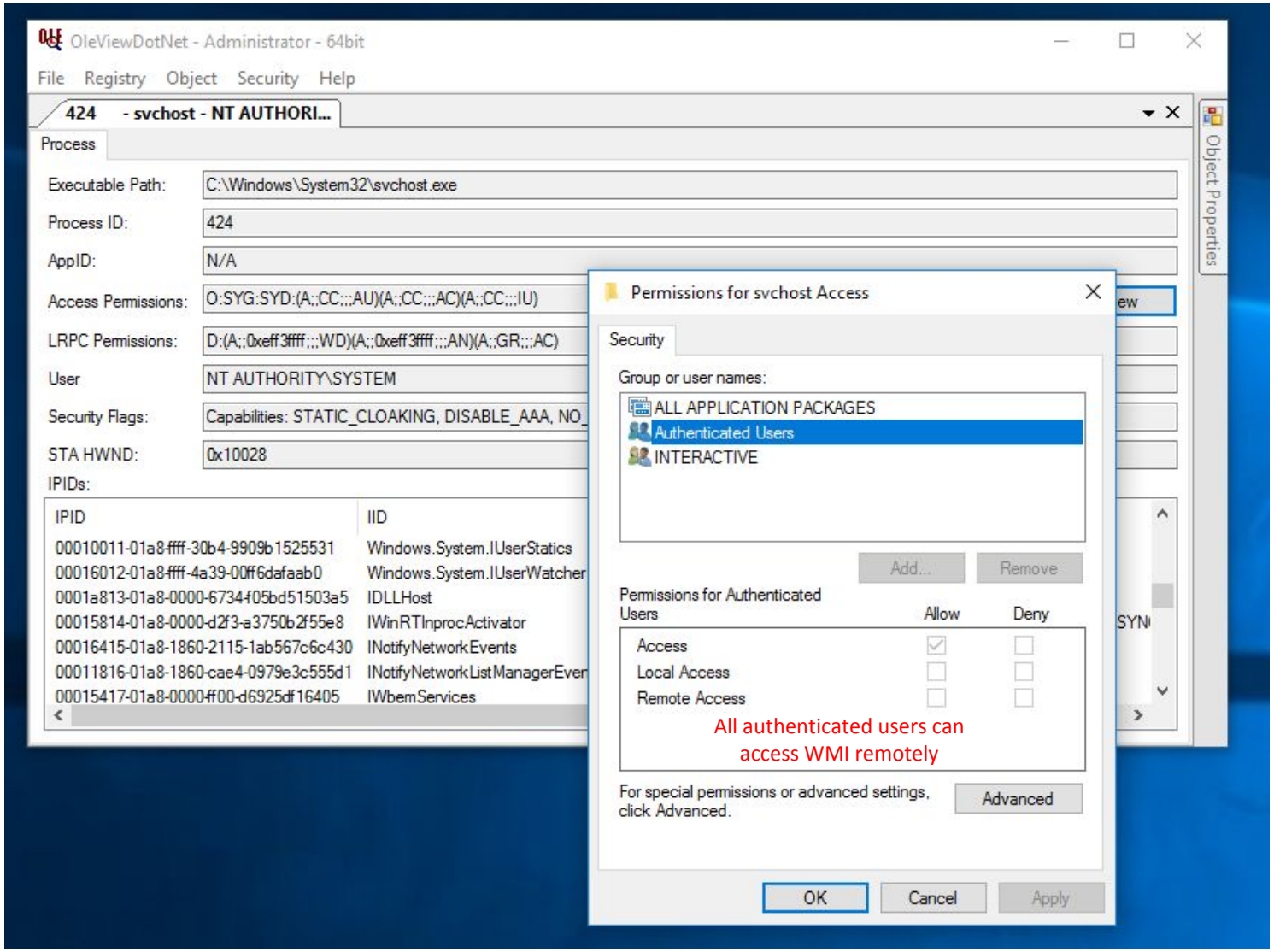
Defaults to CONNECT Authentication Level?

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.

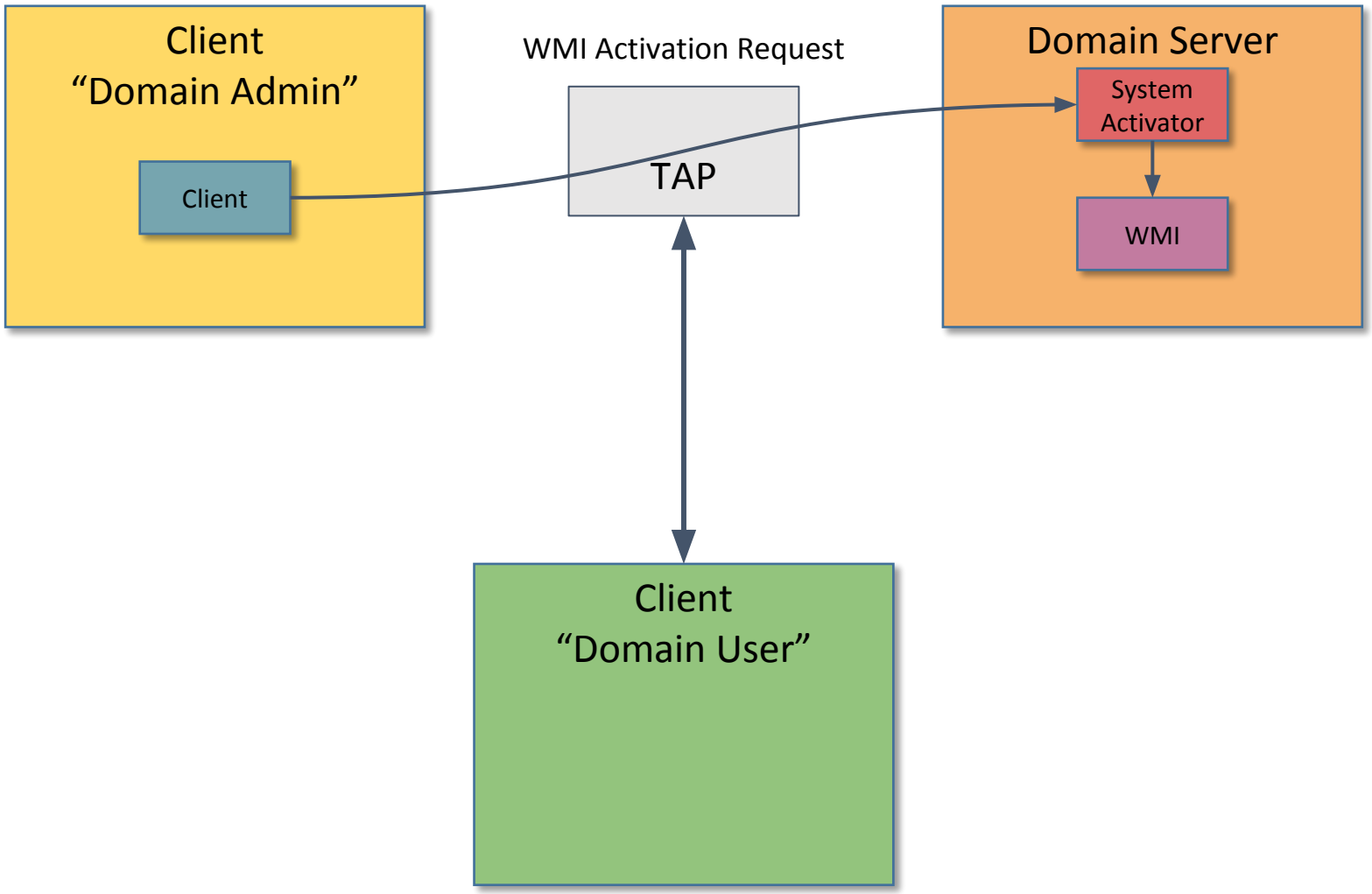
Good old Wireshark



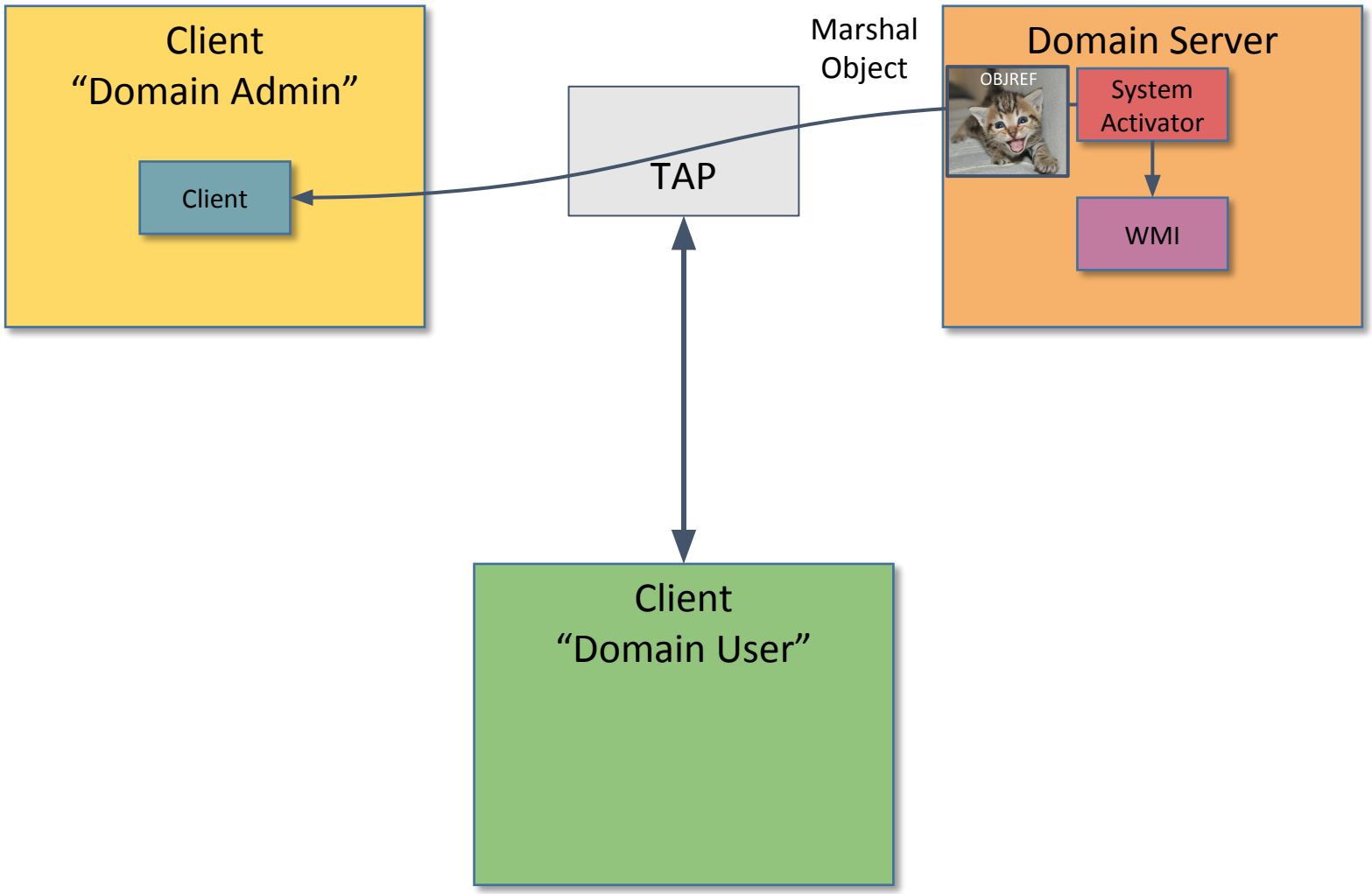
Access Permissions for WMI



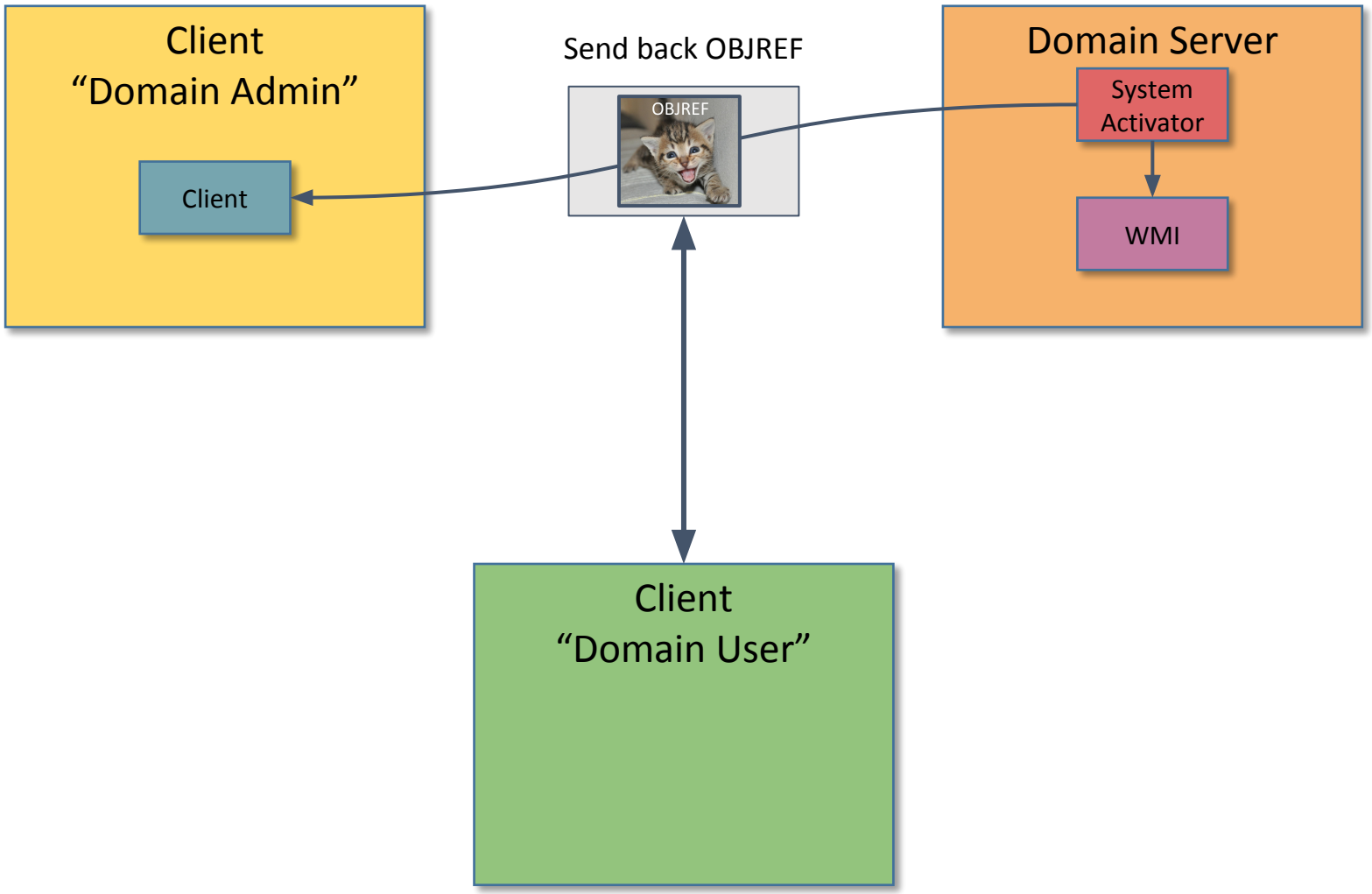
MEOW-Jacking!



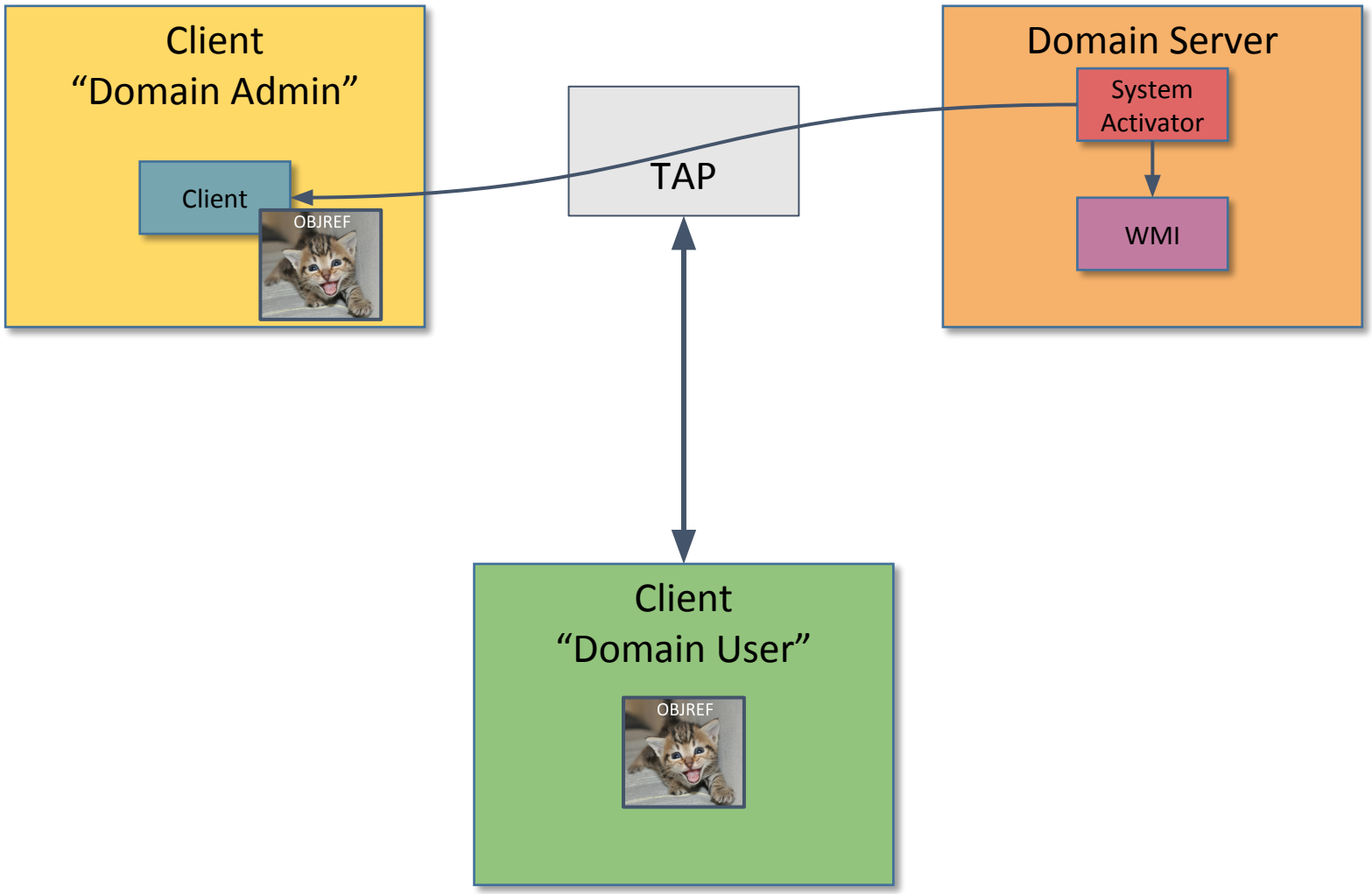
MEOW-Jacking!



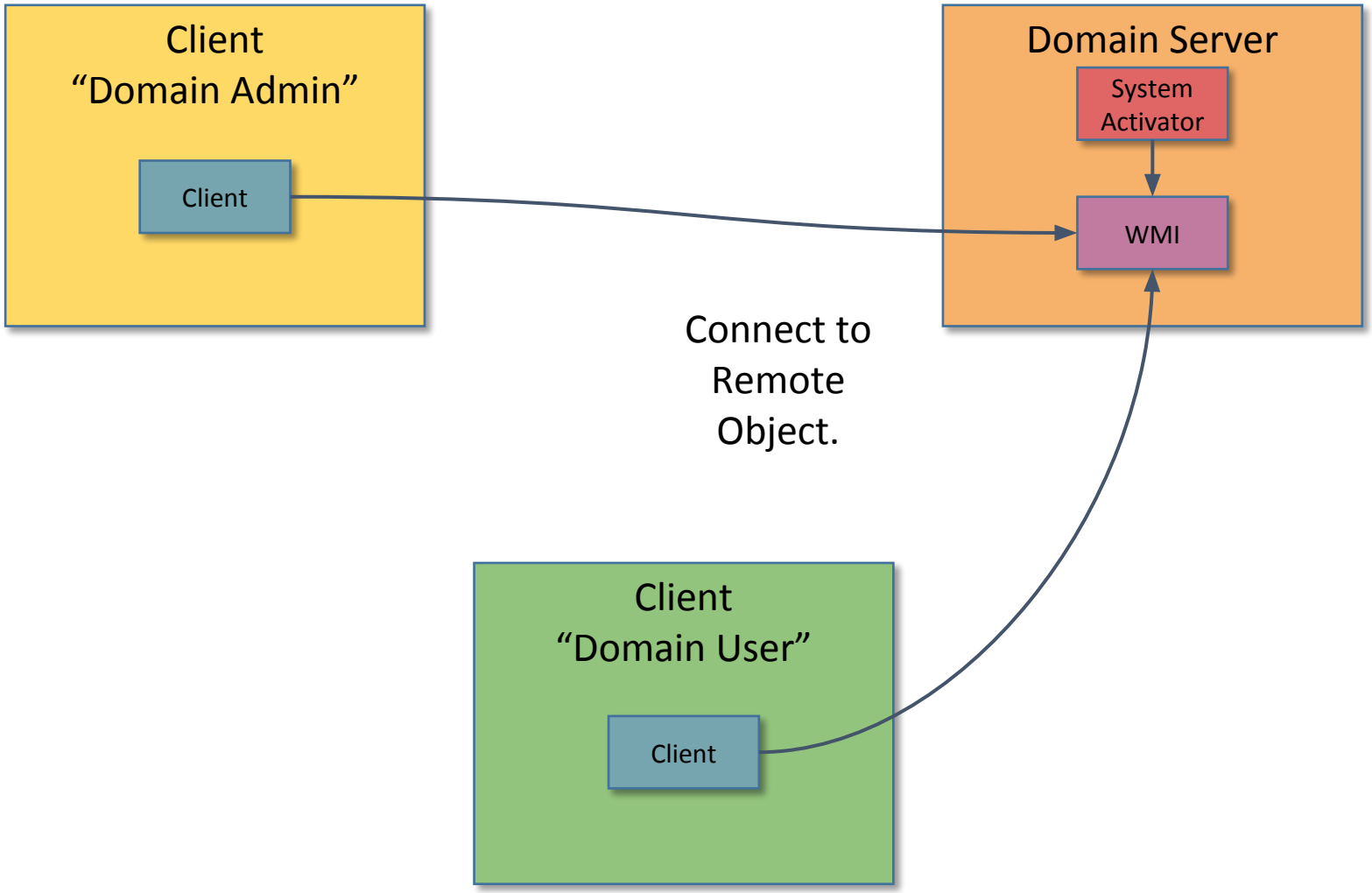
MEOW-Jacking!



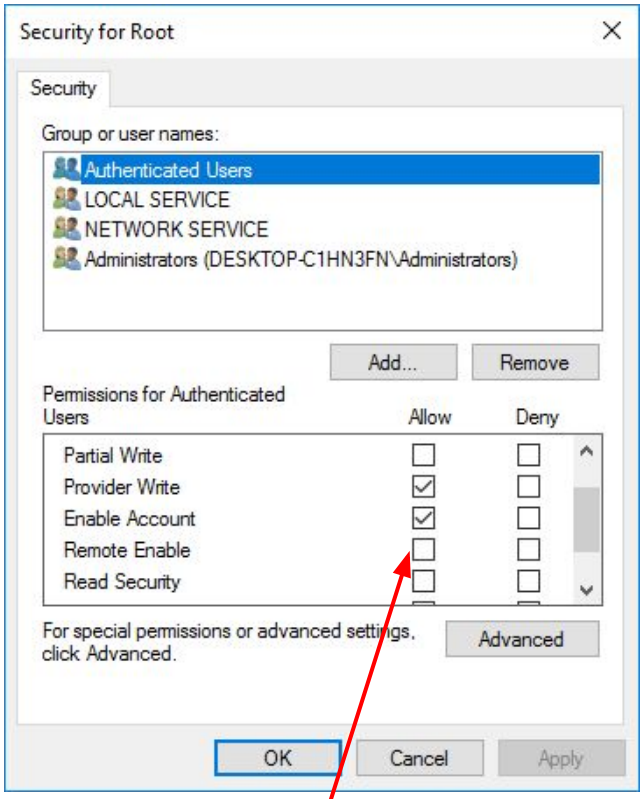
MEOW-Jacking!



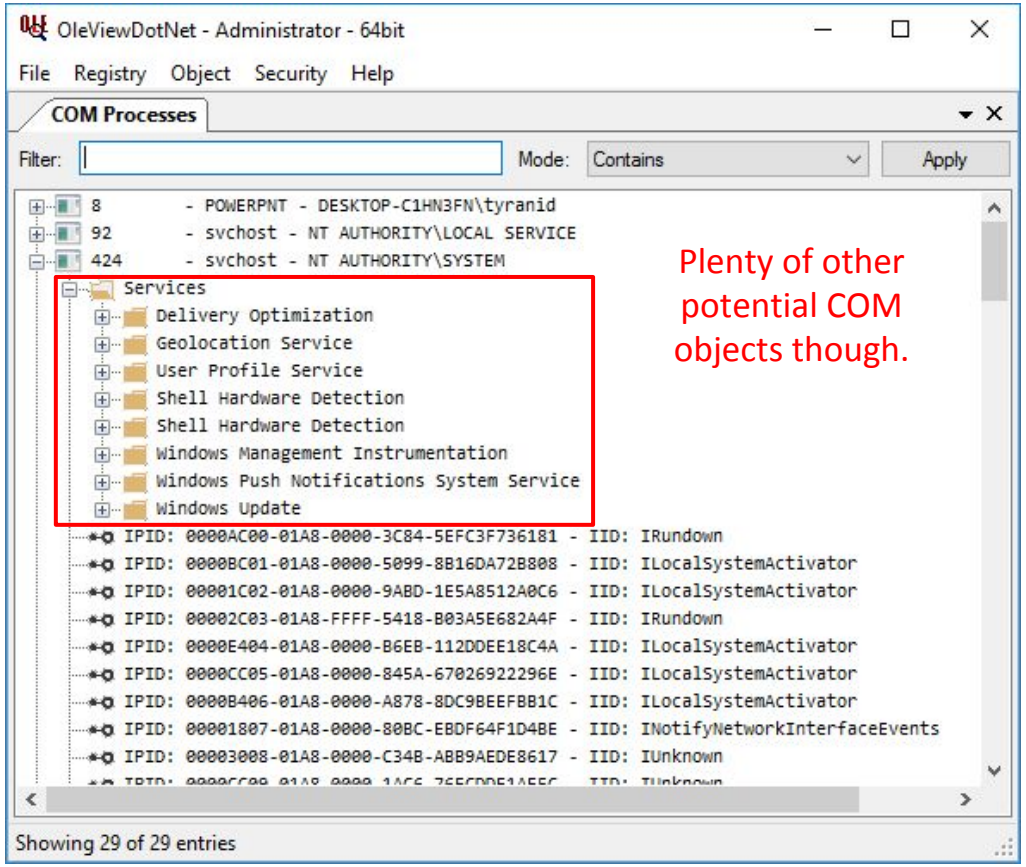
MEOW-Jacking!



Not as Bad as it Could Be



User's not allowed to access WMI remotely.



Plenty of other potential COM objects though.

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

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