WinObjEx64

Windows Object Explorer 64 bit Plugin architecture overview

Document version 1.2.0 (15 June 2025)

Plugins version 1.2.0

Plugins architecture

Implemented as dynamic link libraries (dll).

Dll requirements

Plugin implemented as dll must export "*PluginInit*" (without a quotes) routine. Library must have VERSION_INFO block with *FileDescription* field set to "*WinObjEx64 Plugin V1.2*" (case sensitive, without a quotes). This is used by WinObjEx64 to ensure that given dll is a proper plugin.

Plugin initialization

PluginInit is a WinObjEx64 plugin initialization routine with the following prototype

```
BOOLEAN CALLBACK PluginInit(
    _Out_ PWINOBJEX_PLUGIN PluginData
);
```

Parameters

PluginData – pointer to WINOBJEX_PLUGIN structure that will be filled by plugin. This structure describes plugin and gives WinObjEx64 ability to start/stop it execution.

```
typedef struct _WINOBJEX_PLUGIN {
    ULONG cbSize;
    ULONG AbiVersion;
    union {
        ULONG Flags;
        struct {
            ULONG NeedAdmin : 1;
            ULONG NeedDriver : 1;
            ULONG SupportWine : 1;
            ULONG SupportMultipleInstances : 1;
            ULONG Reserved : 28;
        } u1;
    } Capabilities;
    WINOBJEX_PLUGIN_TYPE Type;
    WINOBJEX_PLUGIN_STATE State;
    WORD MajorVersion;
    WORD MinorVersion;
    ULONG RequiredPluginSystemVersion;
    UCHAR SupportedObjectsIds[PLUGIN_MAX_SUPPORTED_OBJECT_ID];
    WCHAR Name[MAX_PLUGIN_NAME];
    WCHAR Authors[MAX_AUTHORS_NAME];
    WCHAR Description[MAX_PLUGIN_DESCRIPTION];
    pfnStartPlugin StartPlugin;
    pfnStopPlugin StopPlugin;
    pfnStateChangeCallback StateChangeCallback;
    pfnGuiInitCallback GuiInitCallback;
    pfnGuiShutdownCallback GuiShutdownCallback;

    ULONG Reserved[8];
} WINOBJEX_PLUGIN, * PWINOBJEX_PLUGIN;
```

Members

bSize
The size of the structure, in bytes. This should be set to sizeof(WINOBJEX_PLUGIN) by the plugin to ensure versioning compatibility.

AbiVersion
Specifies the ABI (Application Binary Interface) version used by the plugin. Allows the WinObjEx64 application to verify compatibility.

Capabilities
Encapsulates plugin capability flags.

Flags
The raw bitmask value representing various capability flags.

NeedAdmin (bit 0):
Plugin must set to 1 (TRUE) if it requires administrator rights to execute.

NeedDriver (bit 1):
Plugin must set to 1 if it requires a helper driver to function.

SupportWine (bit 2):
Set to 1 if the plugin supports running under Wine (Windows compatibility layer).

SupportMultipleInstances (bit 3):
Set to 1 if the plugin supports being loaded multiple times simultaneously.

Reserved (bits 4–31):
Reserved for future use; must be zero.

Type – WINOBJEX_PLUGIN_TYPE enumeration describing plugin type.

```
typedef enum _WINOBJEX_PLUGIN_TYPE {
    DefaultPlugin = 0,
    ContextPlugin = 1,
    InvalidPluginType
} WINOBJEX_PLUGIN_TYPE;
```

Where *DefaultPlugin* is a general purpose plugins (listed by WinObjEx64 in the main menu Plugins and *ContextPlugin* is the Windows object type specific plugin, they are displayed in the context menu of WinObjEx64 main window when user uses popup menu over selected object in listview or treeview.

State – WINOBJEX_PLUGIN_STATE enumeration describing current plugin state.

```
typedef enum _WINOBJEX_PLUGIN_STATE {
    PluginInitialization = 0,
    PluginStopped = 1,
    PluginRunning = 2,
    PluginError = 3,
    MaxPluginState
} WINOBJEX_PLUGIN_STATE;
```

By default at *PluginInit* this member must be set to *PluginInitialization* (0). Note that *MaxPluginState* is unused.

MajorVersion – Major version field, plugin self defined;

MinorVersion – Minor version field, plugin self defined;

RequiredPluginSystemVersion – Plugin subsystem version that is required by plugin to work. Currently it is 18712 value;

SupportedObjectsIds – An array of plugin supported object types, this field is only valid when plugin type is set to ContextPlugin, for list see. Set *SupportedObjectsIds*[0] to **ObjectTypeAnyType** if plugin intended to work with any object types;

Name – Plugin name, maximum 32 chars (including null terminator), use brief name if possible. The following name is used to identify plugin in WinObjEx64 plugins menu;

Authors – Plugin authors, maximum 32 chars (including null terminator), use brief list if possible;

Description – is a wide char array with maximum size of 128 elements (including null terminator) used to keep plugin human readable description. Use brief description if possible;

StartPlugin – is a pointer to callback routine used by WinObjEx64 to initiate actual plugin work. This field must be set by plugin during *PluginInit* execution.
Prototype defined as following:

```
NTSTATUS CALLBACK StartPlugin(
    _In_ PWINOBJEX_PARAM_BLOCK ParamBlock);
```

The *ParamBlock* is a pointer to WINOBJEX_PARAM_BLOCK structure that will be passed from WinObjEx64 to plugin. Detailed description below;

StopPlugin – is a pointer to callback routine used by WinObjEx64 to initiate plugin shutdown. This field must be set by plugin during *PluginInit* execution.

Prototype defined as following:

```
void CALLBACK StopPlugin(
    VOID);
```

This routine has no parameters;

StateChangeCallback – is a pointer to WinObjEx64 routine that is used to access/modify plugin *State* field. Filled by WinObjEx64, plugins must not modify it.

GuiInitCallback – is a pointer to WinObjEx64 routine that should be called during plugin GUI initialization, it is used to register plugin specific window class.

GuiShutdownCallback – is a pointer to WinObjEx64 routine that should be called during plugin GUI shutdown, it is used to unregister previously registered plugin window class.

Reserved
Reserved for future use.

Remarks

It is advised to make a plugin global variable that reference plugin data during *PluginInit*.

Starting a plugin

WinObjEx64 starts plugins by calling ***StartPlugin*** routine which is set by plugin during *PluginInit* in *WINOBJEX_PLUGIN* structure.

Prototype defined as following:

```
NTSTATUS CALLBACK StartPlugin(
    _In_ PWINOBJEX_PARAM_BLOCK ParamBlock);
```

Parameters

ParamBlock – input parameter, a pointer to *WINOBJEX_PARAM_BLOCK* structure filled by WinObjEx64. Contain pointers to various WinObjEx64 helper routines. Note that structure maybe expanded in the future (growing from tail), for recent version see plugin_def.h in WinObjEx64 Plugins code directory.

```
typedef struct _WINOBJEX_PARAM_BLOCK {
    ULONG cbSize;
    HWND ParentWindow;
    HINSTANCE Instance;
    ULONG_PTR SystemRangeStart;
    ULONG CurrentDPI;
    RTL_OSVERSIONINFOW Version;
    WINOBJEX_PARAM_OBJECT Object;
    pfnReadSystemMemoryEx ReadSystemMemoryEx;
    pfnGetInstructionLength GetInstructionLength;
    pfnOpenNamedObjectByType OpenNamedObjectByType;
    ULONG Reserved[8];
} WINOBJEX_PARAM_BLOCK, * PWINOBJEX_PARAM_BLOCK;
```

Members

Size – the size of the structure, in bytes. This should be set to sizeof(WINOBJEX_PARAM_BLOCK) by the plugin to ensure versioning compatibility.

ParentWindow – is a handle of WinObjEx64 main window;

Instance – is a handle of WinObjEx64 instance;

SystemRangeStart – is a value describing lower possible system start address;

CurrentDPI – DPI value from WinObjEx64;

Version – is a RTL_OSVERSIONINFOW structure which is filled by WinObjEx64 by calling ntdll *RtlGetVersion* function;

Object – is a WINOBJEX_PARAM_OBJECT structure which is filled by WinObjEx64 and valid only for ContextPlugins;

```
typedef struct _WINOBJEX_PARAM_OBJECT {
    LPWSTR ObjectName;
    LPWSTR ObjectDirectory;
    PVOID Reserved;
} WINOBJEX_PARAM_OBJECT, * PWINOBJEX_PARAM_OBJECT;
```

Where ObjectName is the currently selected (in WinObjEx64 treev or listview) object name, ObjectDirectory is a currently browsed object directory. Reserved is a pointer for future use. It is advised to make a local copy of this structure during *StartPlugin* call.

ReadSystemMemoryEx – pointer to WinObjEx64 function used to read kernel memory;

GetInstructionLength – pointer to WinObjEx64 length disassembler wrapper used to determinate instruction length with HDE;

OpenNamedObjectByType – pointer to WinObjEx64 function used to open named objects.

Reserved – reserved for future use.

Remarks

If selected plugin supports multiple instances then new instance of it will be created.

If it does not support multiple instances then if plugin reports in a *State* field that it is already running (*State* is set to *PluginRunning*) then WinObjEx64 will ask user either to restart plugin or leave it as is. In case if user want to restart plugin, WinObjEx64 will first try to stop plugin and then start it again.

Stopping a plugin

WinObjEx64 stops plugin by calling ***StopPlugin*** routine which is set by plugin during *PluginInit* in *WINOBJEX_PLUGIN* structure.

Prototype defined as following:

```
void CALLBACK StopPlugin(
    VOID);
```

This routine has no parameters;

Remarks

Upon successful stop plugin must set state to *PluginStopped* by calling *StateChangeCallback* routine of WINOBJEX_PLUGIN.

Plugin parameters block (WINOBJEX_PARAM_BLOCK)

This parameters block is filled by WinObjEx64. Below is a prototypes of functions within it.

```
BOOL CALLBACK ReadSystemMemoryEx(
    _In_ ULONG_PTR Address,
    _Inout_ PVOID Buffer,
    _In_ ULONG BufferSize,
    _Out_opt_ PULONG NumberOfBytesRead);
```

Read kernel memory to the preallocated buffer.

Parameters

Address – kernel mode address to read;

Buffer – pointer to plugin allocated buffer to receive data;

BufferSize – size of buffer to receive data;

NumberOfBytesRead – optional, return actual number of bytes read upon successful execution.

Return Value

Return TRUE on success, FALSE on failure.

Remarks

Debug privilege and thus administrative rights are required.

```
UCHAR CALLBACK GetInstructionLength(
    _In_ PVOID ptrCode,
    _Out_ PULONG ptrFlags);
```

Length disassembler wrapper.

Parameters

ptrCode – pointer to code;

ptrFlags – pointer to ULONG type variable to receive flags returned by disassembler engine.

Return Value

Return number of bytes describing instruction length of given code buffer.

```
NTSTATUS OpenNamedObjectByType(
    _Out_ HANDLE* ObjectHandle,
    _In_ ULONG TypeIndex,
    _In_ LPWSTR ObjectDirectory,
    _In_opt_ LPWSTR ObjectName,
    _In_ ACCESS_MASK DesiredAccess);
```

Open object by name and type.

Parameters

ObjectHandle – pointer to receive object handle;

TypeIndex – object type index, for index list see;

ObjectDirectory – parent directory of object;

ObjectName – optional, name of the object. If no ObjectName specified function will open ObjectDirectory itself;

DesiredAccess – the access to the object.

Return Value

If function succeeded it returns STATUS_SUCCESS. On failure function return appreciate NTSTATUS value.

Examples

1) Example plugin located in Source/Plugins/ExamplePlugin. Implement basic plugin skeleton. Shows message box as payload.

2) Complex GUI based plugins – Source/Plugins/Sonar and Source/Plugins/ApiSetView.

3) Plugin that support multiple instances – Source/Plugins/ImageScope. This plugin supports "Section" only type of Windows objects.

Object types

Name	Value
ObjectTypeDevice	0
ObjectTypeDriver	1
ObjectTypeSection	2
ObjectTypePort	3
ObjectTypeSymbolicLink	4
ObjectTypeKey	5
ObjectTypeEvent	6
ObjectTypeJob	7
ObjectTypeMutant	8
ObjectTypeKeyedEvent	9
ObjectTypeType	10
ObjectTypeDirectory	11
ObjectTypeWinstation	12
ObjectTypeCallback	13
ObjectTypeSemaphore	14
ObjectTypeWaitablePort	15
ObjectTypeTimer	16
ObjectTypeSession	17
ObjectTypeController	18
ObjectTypeProfile	19
ObjectTypeEventPair	20
ObjectTypeDesktop	21
ObjectTypeFile	22
ObjectTypeWMIGuid	23
ObjectTypeDebugObject	24
ObjectTypeIoCompletion	25
ObjectTypeProcess	26
ObjectTypeAdapter	27
ObjectTypeToken	28
ObjectTypeETWRegistration	29
ObjectTypeThread	30
ObjectTypeTmTx	31
ObjectTypeTmTm	32
ObjectTypeTmRm	33
ObjectTypeTmEn	34
ObjectTypePcwObject	35

ObjectTypeFltConnPort	36
ObjectTypeFltComnPort	37
ObjectTypePowerRequest	38
ObjectTypeETWConsumer	39
ObjectTypeTpWorkerFactory	40
ObjectTypeComposition	41
ObjectTypeIRTimer	42
ObjectTypeDxgkSharedResource	43
ObjectTypeDxgkSharedSwapChain	44
ObjectTypeDxgkSharedSyncObject	45
ObjectTypeDxgkCurrentDxgProcessObject	46
ObjectTypeDxgkCurrentDxgThreadObject	47
ObjectTypeDxgkDisplayManager	48
ObjectTypeDxgkDisplayMuxSwitch	49
ObjectTypeDxgkSharedBundle	50
ObjectTypeDxgkSharedProtectedSession	51
ObjectTypeDxgkComposition	52
ObjectTypeDxgkSharedKeyedMutex	53
ObjectTypeMemoryPartition	54
ObjectTypeRegistryTransaction	55
ObjectTypeDmaAdapter	56
ObjectTypeDmaDomain	57
ObjectTypeCoverageSampler	58
ObjectTypeActivationObject	59
ObjectTypeActivityReference	60
ObjectTypeCoreMessaging	61
ObjectTypeRawInputManager	62
ObjectTypeWaitCompletionPacket	63
ObjectTypeIoCompletionReserve	64
ObjectTypeUserApcReserve	65
ObjectTypeIoRing	66
ObjectTypeTerminal	67
ObjectTypeTerminalEventQueue	68
ObjectTypeEnergyTracker	69
ObjectTypeUnknown	70
ObjectTypeAnyType	0xfe
ObjectTypeNone	0xff