


 This repository Search

Pull requests Issues Gist

gasgas4 / HexRaysCodeXplorer
forked from REhints/HexRaysCodeXplorer

Unwatch 1 Star 0 Fork 85


[Code](#) [Pull requests 0](#) [Wiki](#) [Pulse](#) [Graphs](#) [Settings](#)

Hex-Rays Decompiler plugin for better code navigation — Edit

43 commits 2 branches 6 releases 2 contributors

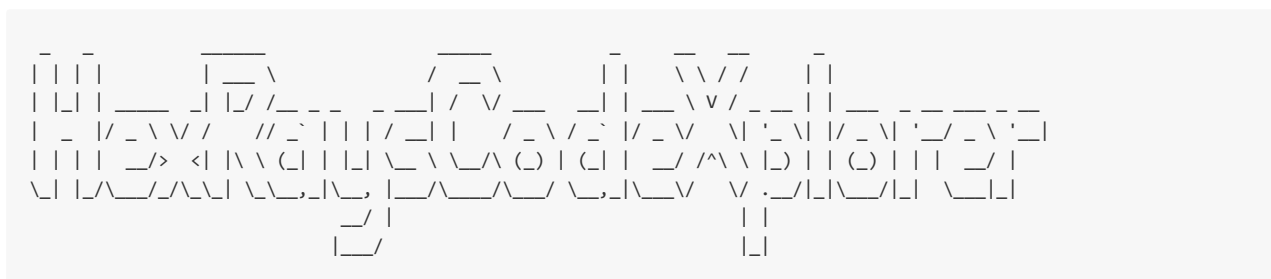
Branch: master New pull request New file Find file SSH git@github.com:gasgas4/HexRay Download ZIP

This branch is even with REhints:master. Pull request Compare

 Alex Matrosov add support for IDA v6.9 and Decompiler v2.3 Latest commit 32cd5e3 9 hours ago

bin	add support for IDA v6.9 and Decompiler v2.3	9 hours ago
img	update	4 months ago
src	add support for IDA v6.9 and Decompiler v2.3	9 hours ago
.gitignore	Initial commit	3 years ago
README.md	Update README.md	4 months ago

README.md



The Hex-Rays Decompiler plugin for better code navigation in RE process. CodeXplorer automates code REconstruction of C++ applications or modern malware like Stuxnet, Flame, Equation, Animal Farm ... 🐱

The CodeXplorer plugin is one of the [first publicly available](#) Hex-Rays Decompiler plugins. We keep updated this project [since summer of 2013](#) and continue contributing new features frequently. Also most interesting features of CodeXplorer have been presented on numerous security conferences like: REcon, ZeroNights, H2HC, NSEC and BHUS 🤖

Contributors:

Alex Matrosov (@matrosov)

Eugene Rodionov (@rodionov)

Rodrigo Branco (@rrbranco)

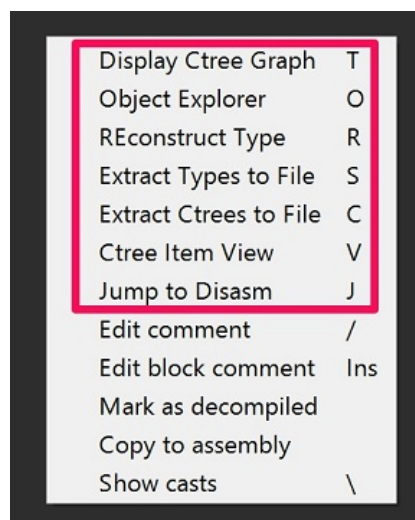
Gabriel Barbosa (@gabrielnb)

Supported versions of Hex-Rays products: everytime we focus on last versions of IDA and Decompiler because trying to use new interesting features in new SDK releases. It's also mean we tested just on last versions of Hex-Rays products and not guaranteed stable work on previous ones.

Why not IdaPython: all code developed on C/C++ because it's more stable way to support complex plugin for Hex-Rays Decompiler.

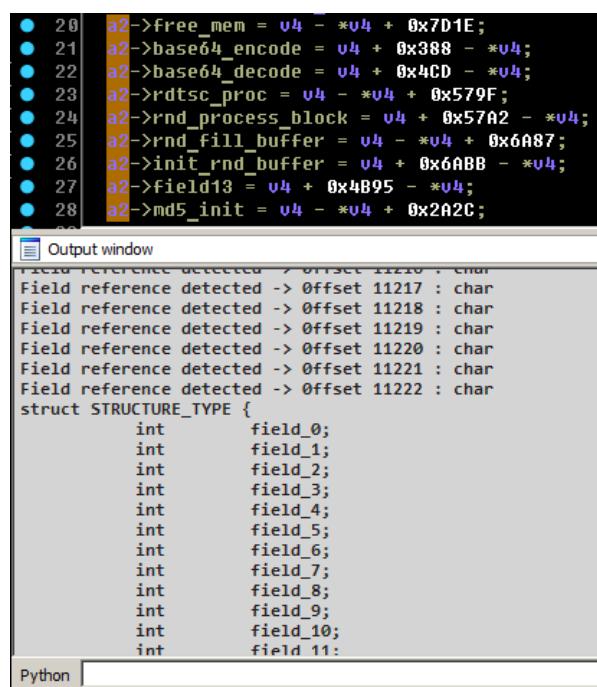
Supported Platforms: x86/x64 for Win, Linux and Mac.

HexRaysCodeXplorer - Hex-Rays Decompiler plugin for easier code navigation. Right-click context menu in the Pseudocode window shows CodeXplorer plugin commands:



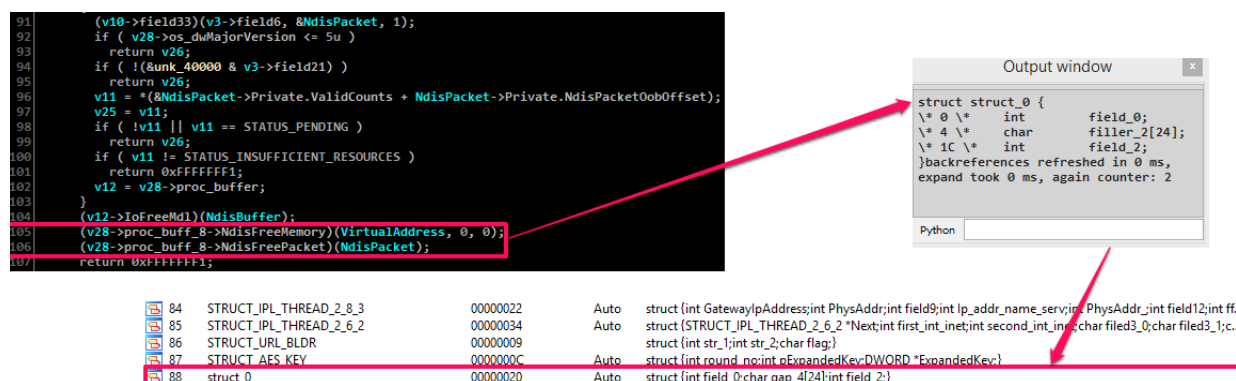
Here are the main features of the CodeXplorer plugin:

- **Automatic type REconstruction** for C++ objects. To be able to reconstruct a type using HexRaysCodeXplorer one needs to select the variable holding pointer to the instance of position independent code or to an object and by right-button mouse click select from the context menu «REconstruct Type» option:

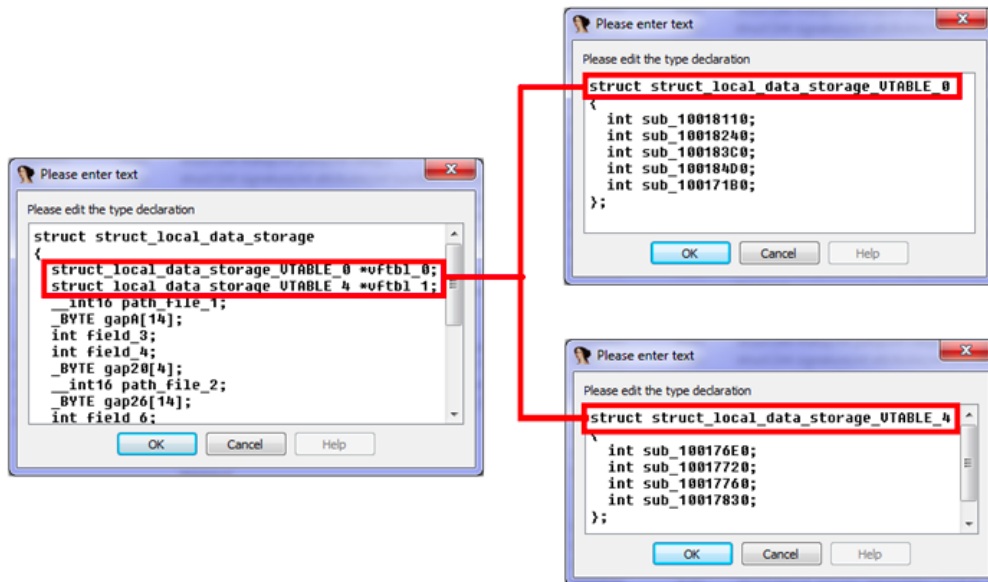


The reconstructed structure is displayed in "Output window". Detailed information about type Reconstruction feature is provided in the blog post "[Type REconstruction in HexRaysCodeXplorer](#)".

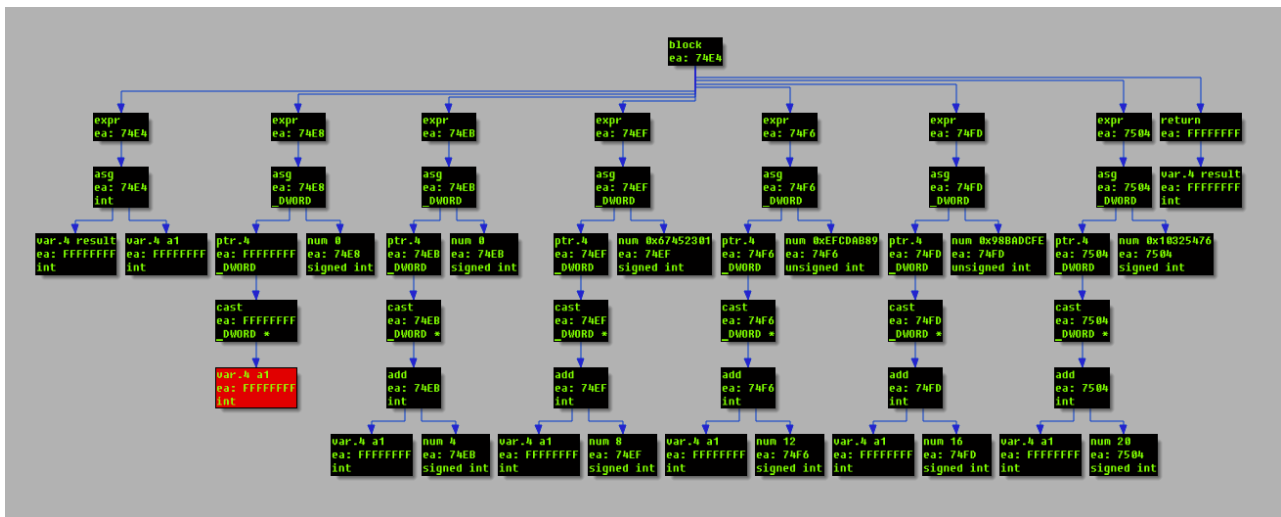
Also CodeXplorer plugin supports auto REconstruction type into IDA local types storage.



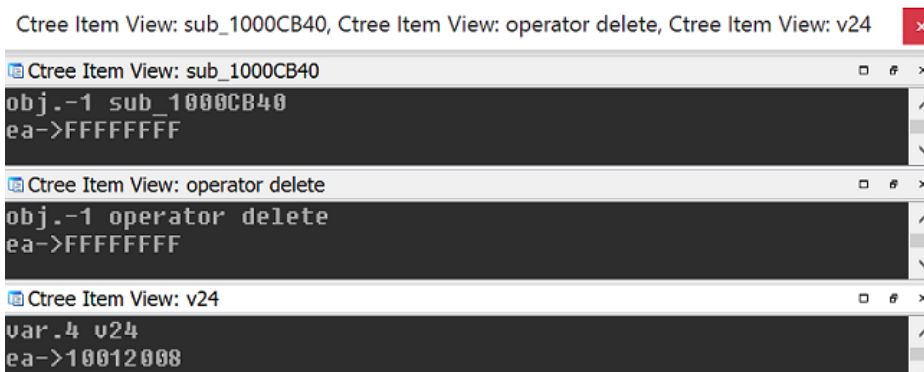
- **Virtual function table identification** – automatically identifies references to virtual function tables during type reconstruction. When a reference to a virtual function table is identified the plugin generates a corresponding C-structure. As shown below during reconstructing `struct_local_data_storage` two virtual function tables were identified and, as a result, two corresponding structures were generated: `struct_local_data_storage_VTABLE_0` and `struct_local_data_storage_VTABLE_4`.




- **C-tree graph visualization** – a special tree-like structure representing a decompiled routine in `citem_t` terms (`hexrays.hpp`). Useful feature for understanding how the decompiler works. The highlighted graph node corresponds to the current cursor position in the HexRays Pseudocode window:



- **Ctree Item View** – show ctree representation for highlighted element:



- **Extract Ctrees to File** – dump calculate SHA1 hash and dump all ctrees to file.

 Please enter a string

Enter prefix of crypto function names

- **Extract Types to File** – dump all types information (include reconstructed types) into file.
- **Navigation through virtual function calls** in HexRays Pseudocode window. After representing C++ objects by C-structures this feature make possible navigation by mouse clicking to the virtual function calls as structure fields:

```

v9 = v8->LowestDevInStack;
if ( v9 )
{
    a2->internal_control_hook_info = (v2->proc_buff_3->hook_routine)(
        0,
        v9->DriverObject->MajorFunction[IRP_MJ_INTERNAL_DEVICE_CONTROL],
        v2->proc_buff_5->IrpMjInternalControl_Hook,
        0,
        0,
        0,
        a2->save_trampoline_info);
    if ( v8->LowestDevInStack->DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] == v8->LowestDevInStack->DriverObject->MajorFunction[15] )
        a2->device_control_hook_info = a2->internal_control_hook_info;
    else
        a2->device_control_hook_info = (v2->proc_buff_3->hook_routine)(
            0,
            v8->LowestDevInStack->DriverObject->MajorFunction[14],
            v2->proc_buff_5->IrpMjDeviceControl_Hook,
            0,
            0,
            0,
            a2->save_trampoline_info);
}

```


- **Jump to Disasm** - small feature for navigate to assembly code into "IDA View window" from current Pseudocode line position. It is help to find a place in assembly code associated with decompiled line.

```

1 int __thiscall GetIterator_0(STR_BLDR_1_STRUCT *this, int a2)
2 {
3     (this->vTable->field2)(a2);
4     return a2;
5 }

```

Display Graph	T
Object Explorer	O
REconstruct Type	R
Jump to Disasm	J



```

mov     eax, offset sub_101C67B1
call    __EH_prolog
push    ecx
and     [ebp+var_10], 0
push    [ebp+arg_0]
mov     eax, [ecx]
call    dword ptr [eax+8]
and     [ebp+var_4], 0
mov     ecx, [ebp+var_C]
mov     [ebp+var_10], 1
mov     eax, [ebp+arg_0]
mov     large fs:0, ecx
leave
retn     4

```

- **Object Explorer** – useful interface for navigation through virtual tables (VTBL) structures. Object Explorer outputs VTBL information into IDA custom view window. The output window is shown by choosing «Object Explorer» option in right-button mouse click context menu:

Object Explorer

6	0x102568b8	- 0x102568c0:	._?AVrhf_yknwaztep@bht@: .?AVdjyynplwo@mzh@	methods count:
7	0x10256928	- 0x1025695c:	off_10256928	methods count: 13
8	0x102569a0	- 0x102569fc:	off_102569A0	methods count: 23
9	0x10256a00	- 0x10256a40:	off_10256A00	methods count: 16
10	0x10256a40	- 0x10256a9c:	off_10256A40	methods count: 23
11	0x10256aa0	- 0x10256afc:	VECTOR_DATA_2_UTABLE	methods count: 23
12	0x10256b48	- 0x10256b68:	off_10256B48	methods count: 8
13	0x10256b68	- 0x10256b88:	off_10256B68	methods count: 8
14	0x10256b88	- 0x10256ba8:	off_10256B88	methods count: 8
15	0x10256ba8	- 0x10256bb0:	off_10256BA8	methods count: 2
16	0x10256bb0	- 0x10256bd8:	FILE_MAPPING_1_UTABLE	methods count: 10
17	0x10256bd8	- 0x10256bf0:	GLOBAL_EVENT_1_UTABLE	methods count: 6
18	0x10267910	- 0x1026796c:	off_10267910	methods count: 23
19	0x10267978	- 0x10267980:	off_10267978	methods count: 2
20	0x102679a0	- 0x102679f0:	PROCESS_HANDLE_UTABLE	methods count: 20
21	0x102679f0	- 0x10267a1c:	off_102679F0	methods count: 11
22	0x10267a1c	- 0x10267a48:	off_10267A1C	methods count: 11
23	0x10267a50	- 0x10267a90:	off_10267A50	methods count: 16
24	0x10267a90	- 0x10267acc:	THREAD_HANDLE_UTABLE	methods count: 15
25	0x10267acc	- 0x10267aec:	off_10267ACC	methods count: 8
26	0x10267aec	- 0x10267af4:	off_10267AEC	methods count: 2
27	0x10267afc	- 0x10267b04:	off_10267AFC	methods count: 2
28	0x10267b08	- 0x10267b7c:	FILE_UTABLE_0	methods count: 29

Object Explorer supports following features:

- Auto structures generation for VTBL into IDA local types
- Navigation in virtual table list and jump to VTBL address into "IDA View" window by click
- Show hints for current position in virtual table list
- Shows cross-references list by click into menu on "Show XREFS to VTBL"

IDA View-A | Pseudocode-B | Object Explorer | Pseudocode-A | Hex View-1 | Structures | Enums | Imports | Exports

29 0x10269b48 - 0x10269bbc: MAIN_VECT_2_HNT_VTABLE methods count: 23
30 0x10269bc8 - 0x10269c3c: MAIN_VECT_2_VOLUME_SUPPLIER_VTABLE methods count: 35
31 0x10269c40 - 0x10269cb4: MAIN_VECT_2_VIRTUAL_VOLUME_SUPPLIER_VTABLE methods count: 43
32 0x10269e10 - 0x10269e84: MAIN_VECT_2_HeadacheConsumer_VTABLE methods count: 36
33 0x1026a008 - 0x1026a064: VECTOR_DATA_1_VTABLE methods count: 21
34 0x1026a068 - 0x1026a080: VECTOR_1_OBJ_1_VTABLE methods count: 22
35 0x1026a080 - 0x1026a098: VECTOR_1_OBJ_11_VTABLE methods count: 23
36 0x1026a098 - 0x1026a0b0: VECTOR_1_OBJ_7_VTABLE methods count: 22
37 0x1026a0b0 - 0x1026a0cc: VECTOR_1_OBJ_1B_VTABLE methods count: 23
38 0x1026a0cc - 0x1026a0e4: VECTOR_1_OBJ_1C_VTABLE methods count: 23
39 0x1026a0e4 - 0x1026a0fc: VECTOR_1_OBJ_8_VTABLE methods count: 22
40 0x1026a0fc - 0x1026a114: VECTOR_1_OBJ_1A_VTABLE methods count: 23
41 0x1026a114 - 0x1026a12c: VECTOR_1_OBJ_12_VTABLE methods count: 23
42 0x1026a12c - 0x1026a144: VECTOR_1_OBJ_14_VTABLE methods count: 23
43 0x1026a144 - 0x1026a15c: VECTOR_1_OBJ_15_VTABLE methods count: 23
44 0x1026a15c - 0x1026a174: VECTOR_1_OBJ_16_VTABLE methods count: 23
45 0x1026a174 - 0x1026a18c: VECTOR_1_OBJ_9_VTABLE methods count: 22
46 0x1026a18c - 0x1026a1a4: VECTOR_1_OBJ_A_VTABLE methods count: 22
47 0x1026a1a4 - 0x1026a1bc: VECTOR_1_OBJ_C_VTABLE methods count: 22
48 0x1026a1bc - 0x1026a1d4: VECTOR_1_OBJ_18_VTABLE methods count: 23
49 0x1026a1d4 - 0x1026a1ec: VECTOR_1_OBJ_17_VTABLE methods count: 23
50 0x1026a1ec - 0x1026a204: VECTOR_1_OBJ_6_VTABLE methods count: 22
51 0x1026a204 - 0x1026a240: STR_BLD_2_VTABLE methods count: 18
52 0x1026a240 - 0x1026a29c: STR_BLD_1_VTABLE methods count: 18
53 0x1026a2a0 - 0x1026a2fc: STR_BLD_3_VTABLE methods count: 18
54 0x1026a31c - 0x1026a334: VECTOR_1_OBJ_3_VTABLE methods count: 22
55 0x1026a334 - 0x1026a34c: VECTOR_1_OBJ_4_VTABLE methods count: 22
56 0x1026a3ac - 0x1026a3c4: VECTOR_1_OBJ_F_VTABLE methods count: 22
57 0x1026a3c4 - 0x1026a3dc: VECTOR_1_OBJ_5_VTABLE methods count: 22
58 0x1026a3dc - 0x1026a3f4: VECTOR_1_OBJ_E_VTABLE methods count: 22
59 0x1026a3f4 - 0x1026a40c: VECTOR_1_OBJ_19_VTABLE methods count: 23
60 0x1026a424 - 0x1026a440: VECTOR_1_OBJ_D_VTABLE methods count: 22
61 0x1026a45c - 0x1026a474: VECTOR_1_OBJ_2_VTABLE methods count: 22
62 0x1026a474 - 0x1026a48c: VECTOR_1_OBJ_10_VTABLE methods count: 23
63 0x1026a4d4 - 0x1026a4dc: STRUCT_2_VTABLE methods count: 16
64 0x1026a4dc - 0x1026a4e8: STRUCT_2_VTABLE_FILE methods count: 21
65 0x1026a560 - 0x1026a58c: RESOURCE_READER_VTABLE methods count: 23
66 0x1026a59c - 0x1026a5ac: MAIN_OBJ_1_VTABLE methods count: 18
67 0x1026aea8 - 0x1026af1c: MAIN_VECT_2_Viper_VTABLE methods count: 25
68 0x1026bd18 - 0x1026bd8c: MAIN_VECT_2_SNAK_VTABLE methods count: 24
69 0x1026bf2c - 0x1026bf4c: STRUCT_6_VTABLE methods count: 16
70 0x1026c038 - 0x1026c0ac: MAIN_VECT_2_PROCESS_NOTIFIER_VTABLE methods count: 36
71 0x1026c208 - 0x1026c34c: MAIN_VECT_2_MONCH_VTABLE methods count: 25
72 0x1026c478 - 0x1026c4ec: MAIN_VECT_2_MICROBE_VTABLE methods count: 25
73 0x1026c4f0 - 0x1026c564: MAIN_VECT_2_MICROBE_SEC_VTABLE methods count: 25
74 0x1026c7d8 - 0x1026c854: MAIN_VECT_2_LUA_RUNNER_VTABLE methods count: 25
75 0x1026ca78 - 0x1026caec: LSSSENDER_VTABLE methods count: 25
76 0x1026ccd4 - 0x1026cce8: MAIN_VECT_2_JIMMY_VTABLE methods count: 25
77 0x1026cce8 - 0x1026cd5c: MAIN_VECT_2_JIMMY_CONSUMER_VTABLE methods count: 34
78 0x1026d010 - 0x1026d034: INET_CONNECTOR_VTABLE_0 methods count: 24

MAIN_VECT_2_PROCESS_NOTIFIER_VT...

0 0x10007ccf: Process_Copy
1 0x1000886f: Process_Init
2 0x10047251: sub_10046BC1
3 0x1004b7e4: sub_1004B7DA
4 0x1009a829: Initialize_PROCESS_NOTIFIER
5 0x1009a947: sub_1009A936

xrefs list

- Support auto parsing RTTI objects:

```

.rdata:100347F4 ; CClassFactory::`RTTI Base Class Array'
.rdata:100347F4 ??_R2CClassFactory@08 dd offset ??_R1A@?0A0A@CClassFactory@08
.rdata:100347F4 ; DATA XREF: .rdata:CClassFactory::`RTTI Class Hierarchy Descriptor'↑to
.rdata:100347F4 ; BaseClass[0]
.rdata:100347F8 dd offset ??_R1A@?0A0A@CClassFactory@08 ; BaseClass[1]
.rdata:100347FC dd offset ??_R1A@?0A0A@CClassFactory@08 ; BaseClass[2]
.rdata:10034800 dd offset ??_R1A@?0A0A@CClassFactory@08 ; BaseClass[3]
.rdata:10034804 dd 0
.rdata:10034808 ; CClassFactory::`RTTI Base Class Descriptor at {0, -1, 0, 4}'
.rdata:10034808 ??_R1A@?0A0A@CClassFactory@08 _RTTIBaseClassDescriptor <offset ??_R0?AUCClassFactory@08, 3, <0, \
.rdata:10034808 ; DATA XREF: .rdata:CClassFactory::`RTTI Base Class Array'↑to
.rdata:10034808 ; 0FFFFFFFh, 0, 40h>; RTTI Base Class Descriptor (#classinformer) ; CClassFactory `RTTI Type De
.rdata:10034820 dd offset ??_R3CClassFactory@08 ; CClassFactory::`RTTI Class Hierarchy Descriptor'
.rdata:10034824 ; const CEventSink::`RTTI Complete Object Locator'
.rdata:10034824 ??_R4CEventSink@06B@ _RTTICompleteObjectLocator <0, 0, 0, offset ??_R0?AUCEventSink@08, \
.rdata:10034824 ; DATA XREF: .rdata:1003480C↑to
.rdata:10034824 ; offset ??_R3CEventSink@08>; RTTI Complete Object Locator (#classinformer) ; CEventSink::`RTTI
.rdata:10034838 ; CEventSink::`RTTI Class Hierarchy Descriptor'
.rdata:10034838 ??_R3CEventSink@08 _RTTIClassHierarchyDescriptor <0, 0, 4, offset ??_R2CEventSink@08>; RTTI Class Hierarchy Descriptor (#classinformer)
.rdata:10034838 ; DATA XREF: .rdata:const CEventSink::`RTTI Complete Object Locator'↑to
.rdata:10034838 ; .rdata:10034874↑to
.rdata:10034838 ; CEventSink::`RTTI Base Class Array'

```

The Batch mode contains following features:

- Batch mode - useful feature to use CodeXplorer for processing multiple files without any interaction from user. We add this feature after Black Hat research in 2015 for processing 2 millions samples.

Example (dump types and ctrees for functions with name prefix "crypto_"):
 idaq.exe -OHexRaysCodeXplorer:dump_types:dump_ctrees:CRYPTOcrypto_path_to_idb

Conference talks about CodeXplorer plugin:

- **2015**
 - "Distributing the REconstruction of High-Level IR for Large Scale Malware Analysis", BHUS [\[slides\]](#)
 - "Object Oriented Code RE with HexraysCodeXplorer", NSEC [\[slides\]](#)
- **2014**
 - "HexRaysCodeXplorer: object oriented RE for fun and profit", H2HC [\[slides\]](#)
- **2013**
 - "HexRaysCodeXplorer: make object-oriented RE easier", ZeroNights [\[slides\]](#)
 - "Reconstructing Gapz: Position-Independent Code Analysis Problem", REcon [\[slides\]](#)

