IDA Pro 5.1 (Feb 2007)

Major Features

MAC OS X DEBUGGER

We have improved Mac OS X support.. IDA now supports fat binaries. Intel Mac OS X applications can be disassembled and debugged. The PowerPC can be disassembled but not debugged. The analysis is enhanced by a new Intel Mac OS X type library (macosx.til). Typical code sequences are often recognized however, because GCC generates highly optimized code, some of them are still missed.

The Intel Mac OS X debugger has the following peculiarities and limitations:

- Mac OS X requires debuggers to have setgid 'procmod' permissions. Please execute the following commands as administrator before using the debugger:

```
chgrp procmod mac_server
chmod g+s mac_server
```

- Hardware breakpoints are not supported. Apparently the operating system does not support them.
- Breakpoints in read-only shared areas are not allowed because they cause system crashes. Therefore, the commands *step over*, *run until return* and similar will fail in those areas.
- Mac OS X emulates UNIX signals on top of the Mach microkernel. The debugger hooks microkernel events to obtain information from the application, effectively hiding the signals from installed signal handlers. This issue might be addressed in a future version.
- In multithreaded applications running on multicore processors, several events may happen simultaneously. For example, if several breakpoints are activated at the same time, IDA will present them sequentially and the application will not resume until the user has dealt with all pending breakpoints.

“USERCALL” FUNCTION CALLING CONVENTION

IDA supports user defined calling conventions. The user may explicitly specify the locations of arguments and return value such as in this example:

```
int __usercall func<ebx>(int x, int y<esi>);
```

which denotes a function with 2 arguments: the first argument is passed on the stack, the second argument is passed in the ESI register and the return value is stored in the EBX register. General rules for the user defined prototypes are:

- the return value must be in a register
- if the return type is 'void', the return location must not be specified
- if the argument location is not specified, it is assumed to be on the stack; the stack locations are allocated consecutively.
- IDA does not check the register sizes but we recommend that you specify correct register sizes (for example, use AL for char type)
- nested declarations are allowed, for example:

```
int **__usercall func16<eax>(int *(__usercall *x)<ebx>
```

```
(int, long<ecx>, int)<esi>);
```

Here the pointer to "x" is passed in the ESI register; the pointed function is a usercall function that expects its second argument in the ECX register, its return value is in the EBX register. The rule of thumb to apply in such complex cases is to specify the registers just before the opening brace for the parameter list.

- registers used for the location names must be valid for the current processor; some registers are unsupported (if the register name is generated on the fly, it is unsupported)
- register pairs can be specified with a colon like <edx:eax>

NORET-ANALYSIS

IDA performs the “no-return” analysis for all functions. It finds out if a particular function returns to the caller or not. This analysis greatly improves the listing quality because many wrong execution paths are detected and truncated at early stages. The user can use the Edit Function dialog box to assist IDA in difficult cases.

This analysis option can be turned on or off in the IDA.CFG file using the AF2_ANORET bit. By default it is active for the x86 processor.

STACK POINTER TRACING BASED ON THE SIMPLEX METHOD

When the number of bytes purged from the stack by many of the called functions is unknown, tracing the stack pointer becomes very difficult. Previous versions of IDA could handle a single unknown call in a function. IDA Pro 5.1 applies linear algebra methods to handle an unlimited number of unknown calls. While this approach is not 100% reliable, it yields correct results in most cases. More about the method can be found here:

https://www.hex-rays.com/blog/simplex-method-in-ida-pro

This analysis option can be turned off in IDA.CFG. The parameter name is AF2_VERSP. At this point, this method of analysis is only available for the x86 family.

EPOC v9

IDA v5.1 adds support for Symbian EPOC v9 files. Both compressed and uncompressed files can be loaded and analyzed.

VISTA

We have added new types from the latest MS Windows SDK. This includes, among other additions, Vista types.

INFINEON Tricore

new processor: Infineon Tricore 32-bit (advanced version)

PROCESSOR MODULES

- PC: **64-bit Visual Studio exception related structures** are recognized.
- PC: 64-bit RBP frames with unusual RBPs are recognized.
- PC: added PC_PROLOG_COLOR/EPILOG_COLOR/SWITCH_COLOR parameters to IDA.CFG
- PC: improved recognition of Borland RTTI data
- PC: improved recognition of jump tables
- PC: Borland frame setup loop for ESP based functions is recognized
- PC: Borland template recognition has been improved
- PC: code cross-references to imported functions are created for imported function pointers
- PC: do not automatically convert the third operand of an instruction into an offset
- PC: when tracing function parameters IDA uses the information that esi, edi, ebp and other registers are preserved by function calls
- PC: keep the stack pointer aligned according to the current segment mode
- PC: **multiple byte nop**'s are supported
- PC: sparse visual studio jump tables are recognized
- PC: **SSE4 (SSSE3)** instructions are supported
- PC: stdcall names like name@8 are used to determine the number of purged bytes
- PC: the type system converts an immediate instruction operand to an offset if it corresponds to a pointer type argument of a function call
- PC: turned on the final pass of the analysis for MS-DOS EXE files
- PC: VxD function calls are recognized in LX/LE files
- PC: support for **Intel VMX instructions** have been added; the undocumented svdc/rsdc instructions are not decoded anymore because they clash with the vmread/vmwrite instructions.
- pc: a reference to seh handler in the form of "push offset handler" can not point to data; if we detect it, we destroy data and create code
- 68K: improved function prolog recognition
- 68K: more jump tables are recognized
- 8051: the opcode FF is considered as insane instruction (to avoid long sequences of FF to be disassembled as code)
- PPC: IDA does not create subi pseudo instructions by default (use PPC_CREATE_SUBI config parameter for that)
- PPC: spoiled register analysis has been improved
- ARM: EPOC thumb glue code is recognized
- ARM: zero value terminates jump tables which follows the LDR PC instruction
- ARM: IDA knows that ldr pc, sp... can not be a table jump insn
- ARM: recognize disguised function calls like "mov lr, pc; add pc,..."
- Palm: many changes for better analysis of Palm programs; added support for ARM Palm applets (thanks to Igor Skochinsky)

FILE FORMATS

- **COFF: added support for new big COFF files from Microsoft**
- COFF: COMDAT segments are marked in the disassembly
- COFF: files with the huge number of segments can be loaded
- COFF: TMS740 machine type has been added (thanks to Job de Haas)
- AOF: ignore aux names because they hinder creation of data items in the disassembly
- AOUT: dynamic symbol table is parsed and used in the disassembly

- ELF: several new ARM relocations are supported
- NLM: IDA knows about prelude and termination functions and automatically loads NLM.TIL
- OMF: Digital Mars OMF extensions are supported
- OMF: Digital Mars SECREL TLS relocation type is supported
- PDB: consider names starting with _IID_ as GUIDs
- PDB: PDB plugin can be turned off from the command line: -Opdb:off
- PE: information from the demangler is used to detect data names in the code segments
- PE: files with zero entry point lead to the automatic loading of the file header
- PE,XBE: added vcseh.sig file to detect SEH_prolog/epilog functions even if the input file has no imports
- TDS: added a workaround for invalid sstNames sections
- TDS: added support for esp based variables. unfortunately ilink32 does not always set the corresponding bit
- TDS: IDA understands TDS information embedded into the executable file
- Mach-O: non-trivial segments without sections are automatically loaded (this **reveals hidden code**)
- Mach-O: **x86 relocation support** has been added; some relocation types are not handled yet
- Mach-O: **fat (multiple architecture) files** are supported

KERNEL

- much better stack pointer tracing for IBM PC
- 2 new offset attributes have been added: RVA offsets and 'past object'
- abstract (nameless) declarations are accepted in the 'set type' command
- added -C# command line option to specify the compiler used to compile the input file
- added an option to turn off data coagulation in code segments
- added ids files for 64bit MFC
- added MFC8 IDS files
- BDS signature files have been updated (Thanks to Peter Sawatski!)
- BDS4 signature file, contribution of an IDA user
- Borland RTTI recognition has been improved
- “create dref32” rule has been improved: now it verifies that the conversion result is representable in the listing
- demangler information is now used to set up the current compiler
- Digital Mars compiler signatures have been added
- final autoanalysis pass is more careful with the alignment instructions
- FLAIR: it is possible to specify the compiler in the startup signatures using the c=# sequence
- FLAIR: TLS related x86 relocations are supported in the PELF utility
- IDA can guess the type of functions which use floating point arguments
- IDA does not automatically create unicode strings in ms dos programs
- IDA does not delete the user defined names when creating other items
- IDA performs additional analysis to determine if functions return to their callers
- IDA uses demangler information to guess function types
- IDS files can be applied to the database of the DLL itself
- IDS: many MS Windows IDS files have been updated
- if the instructions at the end of a function are deleted, IDA automatically truncates the function
- in order to load COFF/OMF/NE files with huge number of segments, IDA fills the gaps between the segments with .align directives

- jump table targets are never converted to functions
- Mac OS X type library has been added
- name demangling autodetects compilers (IDA_NOAUTOCOMP environment variable can be used to revert to the old behaviour)
- new Borland VCL signature
- new keyword in IDT files: Exit. It is used to denote functions which do not return
- OpenWatcom v1.5 signatures have been added
- repeatable comments from the base address of a complex offset expression are skipped to give place for the comments coming from the offset target (requires reanalysis)
- replaced block characters by -+| characters in the function borders and listing header
- segment type directly depends on the segment class. for example, if the segment class is specified as "DATA", the segment type will be SEG_DATA
- type of unicode constant strings is guessed correctly (as wchar_t)
- user specified zero stack modification are not overwritten by IDA
- uunp, chartbox: removed the hotkeys for these plugins because they could hinder their use for windows switching
- Visual Studio signature files have been updated
- zero length bitfields in c declarations are allowed and ignored
- the analysis is faster in many cases

IDC & SDK

- IDC: Added __GUI__ and __TXT__ predefined macros. One of them will be defined depending on the current user interface
- IDC: added functions to handle breakpoints
- IDC: added functions to query and manipulate function chunks (tails)
- IDC: defined the SEARCH_UP symbol to denote search backward
- IDC: new function ApplySig() to load flirt signatures
- SDK: added the qflow_chart_t class for more control on flow charts
- SDK: added a callback for the graphs: grcode_destroyed. It is generated when the displayed graph is destroyed.
- SDK: added a callback to preprocess flow charts before displaying them in the graph view (processor_t::preprocess_chart)
- SDK: added a hotkey parameter to viewer_add_menu_item()
- SDK: added add_qword() to facilitate 64-bit relocation application
- SDK: added AFL_FIXED_SPD flag to denote instruction with fixed stack pointer delta. This flag is set at the instruction which has a fixed effect on SP (CORRECTION: it is applied to the address past the instruction)
- SDK: added callbacks to generate custom hints
- SDK: added decode_preceding_insn() to decode instruction which precedes the specified address in the execution flow
- SDK: added eavec_t and intvec_t convenience typedefs
- SDK: added events for function creation, deletion, and boundary modification
- SDK: added FUNC_SP_READY flag to denote functions with SP analysis performed; removed ua_emu() and made automatic reanalysis of new instructions; added verify_sp notification to perform sp-analysis
- SDK: added function_item_iterator::decode_preceding_insn() to decode instruction in the reverse execution order
- SDK: added get_current_thread() function
- SDK: added get_funcarg_size() to calculate the size of a function argument

- SDK: added helper alignment template functions align_up and align_down
- SDK: added non-standard functions like add_unique() and del() to the qvector template
- SDK: added possibility to specify custom navigation band colorizer
- SDK: added PR_ALIGN_INSN for some processors; this flag allows IDA to create 'alignment' instructions (e.g. nop) arbitrarily. In general it is a bad idea because such an instruction may spoil the listing by triggering the creation of other wrong instructions
- SDK: argloc_t definition has been modified to hold more information and be easier to work with; the old definition is still available if the NO_OBSOLETE_FUNCS is not defined
- SDK: build_funcarg_arrays() converts array arguments into pointers
- SDK: do_unknown() and do_unknown_range() accept flags as the last parameter instead of a simple bool. This allows us to specify more options. One new option: DOUNK_DELNAMES will delete the names in the specified range
- SDK: FUNC_PURGED_OK flag has been introduced to mark functions which certainly do not modify the stack; sp-analysis is improved to use this flag
- SDK: gdl_graph_t has 2 more virtual functions: get_node_color and get_edge_color
- SDK: get_data_type_size_by_flags() is optimized not to access the database if not really necessary
- SDK: get_data_type_size_by_flags() is renamed as get_data_elsize()
- SDK: get_off_expr and get_name_expr functions accept one more flag: GETN_NODUMMY. This flag can be used to verify that the expression can be built without really creating dummy names in the database
- SDK: get_prev_func_addr() and get_next_func_addr() functions have been added. they return the previous/next address in the function strictly linearly
- SDK: graph interface has been changed to use the qvector template. It is with accessible from other compilers (was only from BCB6)
- SDK: if expand_struc() does not change anything in the structure definition because the offset is bigger than the structure size, it is not considered as an error
- SDK: introduced a bit for user defined type information attached to program items (AFL_USERTI). Types derived by IDA will not have this bit.
- SDK: introduced a flag for 'alignment' instructions: is_align_flow(). It is used to denote control flow created only because of an alignment instruction
- SDK: introduced the qtype file to keep type information in the dynamic memory
- SDK: is_type_void() now checks if the type is really 'void'; before it was reporting 'true' on partial types based on the BT_VOID symbol; the old version of this function is still available if the NO_OBSOLETE_FUNCS symbol is not defined at the compilation time
- SDK: made graph interface compatible with compilers other than BCB6; for that, removed the bn memory of the flow_chart_t class
- SDK: new event 'determined_main' is generated when the main() function of the application is determined
- SDK: new event: loader_elf_machine to handle elf files of unknown machine types
- SDK: new event: treat_hindering_item. This event allows plugins to handle the situation when an old item hinder the creation of a new item (e.g. a wrong data item prevents the creation of an instruction)
- SDK: new function add_auto_stkpnt2() which replaces add_auto_stkpnt(). The old function should not be used anymore since it does not take into account the function boundaries
- SDK: new function add_til2() with more flags
- SDK: new function calc_switch_cases() to calculate switch case values and targets; jptcmn.cpp file has been modified a lot: if you used it, then take the old version from sdk5.0 or update your module
- SDK: new function get_func_nargs(). It returns the number of arguments if the given

function type
- SDK: new function make_array_type() to create array types
- SDK: new function qlgetz() to read zero terminated ascii strings from the input stream
- SDK: new function set_compiler() to set the current compiler
- SDK: new function set_purged() to specify a new value for the "purged bytes" attribute and reanalyze the relevant functions
- SDK: new notification event: processor_t::renamed. It is generated when a new name is given to a byte
- SDK: print_type_to_one_line() can be called with buf==NULL and bufsize==0 to find out the required buffer size
- SDK: QASSERT() and interr() debugging helpers are added
- SDK: QPRM() macros are useable for plugins
- SDK: qstring template class has more member functions
- SDK: removed HT_GRAPH and added 2 parameters to create_graph_viewer() This breaks the existing graph view plugins (sorry) but this part of IDA API has not been settled yet.
- SDK: set_ti() and set_op_ti() always update the field list in the database
- SDK: switch_info_t structure has been improved to handle more cases
- SDK: tform interface was unusable because the window handles were not valid at the form creation time; added events tform_visible/tform_invisible to solve this problem
- SDK: the kernel handles CM_CC_SPECIAL calling convention; the ph.calc_arglocs callback does not need to handle it anymore
- SDK: the node titlebar height can be specified in create_graph_viewer()
- SDK: ua_dodata2() replaces ua_dodata(). The new function handles correctly references to external symbols. ua_dodata() could not handle them because it had no operand offset information and could not calculate the base address of the name
- SDK: ua_outop2() allows to specify GETN_NODUMMY ans obtain text representation of an operand without modifying the database;
- SDK: ua_stkvar2() function accepts 'flags' parameter. Currently there is only one constant: STKVAR_VALID_SIZE. It means that the x.dtyp field contains the stack variable data type. For instructions like "lea" this flag should be cleared.
- SDK: added get_bg_color callback: plugins can dynamically change background colors
- SDK: pc: added save/restore pushinfo callbacks for function prolog information
- SDK: new function dummy_name_ea() to detect dummy names
- SDK: parse_types2() allows to explicitly specify the pa

USER INTERFACE

- gui: the middle mouse button closes tabs
- gui: the maximal number of graph nodes can be changed by the user
- graph overview displays nodes in color
- gui: reuse the existing stack view on Ctrl-K
- -T command line switch to specify the file type of the input and to hide the 'load file' dialog box
- new environment variable IDALOG_SILENT to turn off the messages in the message window; if set, the messages will only be logged to the file specified by IDALOG
- txt: slightly increased the keyboard cursor size (15%, was 10% of the cell)
- ui: ida proposes the last used alignment as the default one for the "make alignment" directive
- ui: the patch submenu is visible under the debugger regardless of the config file settings
- ui: try to create c style strings too if the current string type is pascal and we could not create it when the user requested a string creation

DEBUGGER

- debugger: Mac OS X debugger has been added
- debugger: names exported from a dll can be displayed by double clicking in the module list
- debugger: it is possible to jump to a location like "fs:0"
- debugger: optimized breakpoint insertion/deletion (huge breakpoint list would slow ida down tremendously)

BUGFIXES

- 64-bit and 32-bit user interface data defaults are incompatible with each other and should not be stored in the same registry key uiConfig
- 64-bit version could erroneously check files with IDB extension before displaying questions like "old database will be overwritten" and similar
- a wrong function could be marked as a 'thunk' function
- added support for SSE3 instructions (x86)
- Alt-K is removed from the hotkey list in some more dialog boxes
- AMD64 COFF relocations were not taking into account the displacements
- an unpacked corrupted (or old) database could be deleted by ida if the user told ida that he wants to use the unpacked version but does not want to repair (or upgrade) it
- apparently the Windows kernel does not use the FileAlign field of the PE header if the object alignment is equal to the page size; IDA was always using it which was wrong
- ARM: some forms of the MSR instruction were not disassembled
- calc_stkvar_struc_offset() could return a value with inverted sign
- char array[sizeof(void*)] could not be parsed
- command line plugin options were not available for PLUGIN_PROC plugins
- const and volatile modifiers of structure member declarations were not always processed correctly (nested const declaration of a structure would fail)
- corrupted graph group information could cause IDA to issue a fatal error message and exit
- create_flow_chart() could create a wrong edge if the last node of the flowchart was passing execution control to the next instruction; also external nodes with several references to them would be duplicated
- DEL_CODE_COMMENTS ida.cfg parameter was broken
- del_frame() was clearing number of purged bytes
- delayed import table was incorrect for 64 bit programs
- deleting all stack change points could lead to wrong result of get_spd()
- dump to idc: if there were several enum members with the same value, all comments were attached to the first enum member
- ELF loader would use symbols from skipped (not loaded) sections
- expand a structure type could overflow its size and corrupt the type definition
- functions returning pointers to arrays could not be specified
- get_name_value() could erroneosly return NT_BYTE if a spurious netnode with the specified name was existing
- get_prev_member_idx() was incorrect
- get_true_name() with the buffer size <= 1 would delete the name
- graph view hotkeys were not working without a right-click
- guessed function types without any arguments would explicitly have nargs=0 while it was not necessary
- hex views were not refreshed during debugging
- IDA could crash if a group node had too long text
- ida could crash trying to determine a data item type if it was a self-referencing offset

- IDA could crash while working with the watches in the debugger
- IDA could create wrong offet operand for lea instruction, causing incorrect xrefs
- IDA could cycle endless in the batch mode if the idb file could not be created for some reason
- IDA could delete a wrong function in some circumstances (user-interface "Delete function" is invoked, Function deletion confirmation is on, Background analysis is running, The database has many functions, User does not respond to the confirmation request immediately)
- IDA could delete some imported names at the loading time
- IDA could hang on PE files with corrupted import table
- IDA could hang trying to guess the function type if the stack frame was bigger than 0x80000000
- ida could not demangle some borland names (extremely rare cases)
- IDA was proposing Zilog8 COFF file format as the second choice for some Window PE files
- IDA would continuously load/unload debugger plugins if the current file was not using them
- IDA would not always reanalyze function tails after modifying the stack pointer
- IDA would not recognize some labels in the 'jump to' dialog box
- ida would silently fail to open too small databases (zero size, for example)
- IDC command line was not working
- IDC: INF_AF bit was erroneously named INF_START_AF
- if an instruction overlapping a function tail begining is created, the tail header would not be visible; now we delete the tail in this situation
- if the user asked to invert the sign of a zero displacement, the minus sign would be missing from the output
- in some cases the term() function of a plugin would not be called
- in the graph handling calc_fitting_params() had a bug - i'm not sure of its consequences and if it was visible for the end user
- instructions added by processor extensions would always have coprocessor number
- invoking small help window from dialog boxes would display the help page but would also display 'help not found' error message
- is some rare cases IDA could crash trying to create instructions at the addresses with existing data items
- is_castable() was rejecting some type pairs as incompatible
- it was impossible to prohibit kernel actions from a notification callback in a plugin
- it was not possible to create many individual elements using the array command in special segments
- it was not possible to quit IDA after detaching from a process using a mini-database if the idagui.cfg ASK_EXIT and ASK_EXIT_UNPACKED were set to NO
- java module could not handle a double-slot local variable overlapped with a single-slot variable
- mach-o loader could hang on some input files trying to parse __cat_inst_meth and similar sections (thanks to Jeremy Cooper for this bugfix!)
- MIPS processor module was not using delayed branch information to form basic blocks
- moving a segment could make inaccessible the file offset information which is displayed on the status bar
- nextthat/prevthat with f_isUnknown were not working under the debugger
- on dual monitor mode, the graph overview window would be displayed on the monitor with the main ida window; the same behaviour could be observed with some other windows
- pc: data type specifiers of some simd/sse instructions were wrong

- pc: IDA was taking into account the 'purged bytes' parameter even for non-returning functions
- pc: IDA was using "qword ptr" instead of "fword ptr"
- pc: scas and cmps feature bits were wrong; these instructions do not modify their operands
- Philips 51XA module was missing some code references
- Philips 51XA: stack tracing was incorrectly setting the modification points too early
- PowerPC module was too aggressive with the offset creation
- pressing K would not automatically create a stack variable for the current operand in some cases
- printable plugin hotkeys could not be used in the notepad
- punctuation colors in ascii directives were incorrect
- REFINFO_NOBASE was not available in IDC
- removed some screen flicker when jumping to a non-graph location while in the graph preference mode
- right clicking on an operand does not create spurious dummy names anymore
- SH3 delay slots were not displayed correctly in the graph view
- SH3: the mova instruction does not create a 'read' reference but an 'offset' reference
- some thunk functions were not marked as thnk (if the jump target had no name and the thunk itself had a name)
- special segments (extern, imem) were displayed incorrectly in the address lists (like xref list)
- spurious xrefs from special segments (extern, imem) could be created
- text version was behaving differently from the gui version regarding operation cancellation using Ctrl-Break (the flag was not cleared when it sould have been)
- text version was not saving the text representation options in the database correctly
- the calculation of the number of recognized function in flirt was not always correct
- the comment how to use dumped IDC files was wrong
- the current location indicator was not refreshed after left/right arrows in the hex view
- the debugger could miss some segments on some linuxes
- the debugger server was crashing if the connection could not be established for some reason
- the default button name was always displayed as 'cancel' in the batch mode log
- the length of ascii strings was calculated without taking into account the color code bytes
- the maximal number of marked location was 1023, not 1024 as declared in the header file
- the notepad contents were not saved by the 'save' command
- the presence of duplicate symbols in an elf file would lead to a complaint about unusual relocation items
- the screen was not refreshed after running an idc script by clicking on the script toolbar
- the structure type footer was not displayed correctly for structures with sizeof() == 0xFFFFFFFF
- thiscall calling convention clears stack; ida was not taking this into account
- TIL files for Borland BCB are compatible with Delphi and accepted silently even if the current compiler is Delphi
- til files were not freed at the 'close' command
- tms320c3: the emulator could erroneously display a warning about absi and bsf instructions
- tms320c54 stack variables were wrong
- tms320c54: function frame size was not determined correctly
- txt: in the 'text representation' dialog the alt-k hotkey for the stack value display was not avaiable
- unknown element types in .Net files could cause an error message followed by a crash
- unknown type signatures were causing garbage bytes in the output for .net files
- when editing 64bit user-defined offsets, the default offset type was wrong

- when ida window was resized to an odd size, it could takes very long time to refresh the screen
- when specifying 64-bit user-defined offsets IDA was asking for the target address which was not necessary
- WinCE debugger could miss some segments because they were not physically present in the memory at the inquiry time; now it pretends that the program segments are always present (even if they are not)
- ARM: IDA was confusing some plain branches with tail calls
- ascii string style could not be set from the general options dialog box
- x86 smsw instruction had wrong data types
- ascii string style could not be set from the general options dialog box
- IDA could crash if a function was appended as a tail to another function
  idb structure definitions with 'pointer to function' members could not be converted into type strings
- it was impossible to change navigation band colors from the colors dialog box
- qisabspath() was wrong
- sam8 processor module was setting black segment color making the listing difficult to read
  some MS DOS MS overlays could not be loaded
- typedef enum {...}; would be rejected as wrong type (in fact it is wrong but compilers allow it)
- z80 assemblers use parenthesis to indicate memory access; ida was using them for any complex expression
- "disassembly memory organization" dialog box could automatically resize the RAM or ROM segments thus confusing the user
- 64bit: the 'search all' results window had the high 32-bits of all addresses truncated
- basic block boundaries were not always correctly calculated
- fatal registry related errors could cause a crash
- launching ~20 copies of IDA simultaneously could lead to registry related errors
- OMF files with dummy names could not be correctly loaded
- PC module could crash in some very rare cases
- pe: the size of the delayed import address table size was too big
- when commenting VxD calls we could spoil the listing

Contact Information

© 2007 DataRescue SA/NV

40 Blvd Piercot

4000 Liège

Belgium

t - +32-4-3446510

f - +32-4-3446514

info@datarescue.com