

Diffing and Merging Databases with IDA Teams

Last updated on May 25, 2023 – v8.3

1. Overview

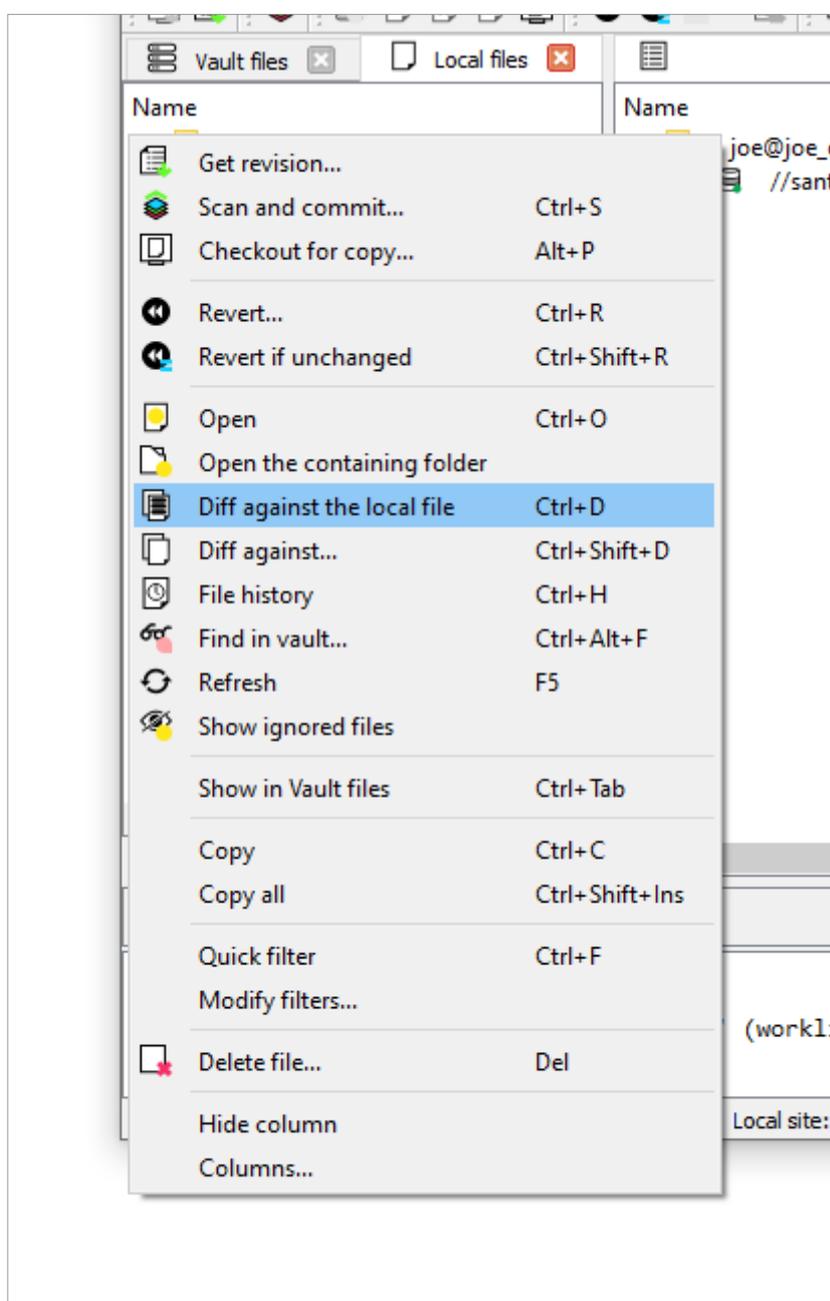
IDA 8.0 introduces IDA Teams - a mechanism that provides revision control for your IDA database files. Perhaps the most essential feature of this new product is the ability to natively diff and merge databases using IDA, allowing multiple reverse engineers to manage work on the same IDA database.

This document discusses in detail the steps involved when diffing and merging IDA databases.

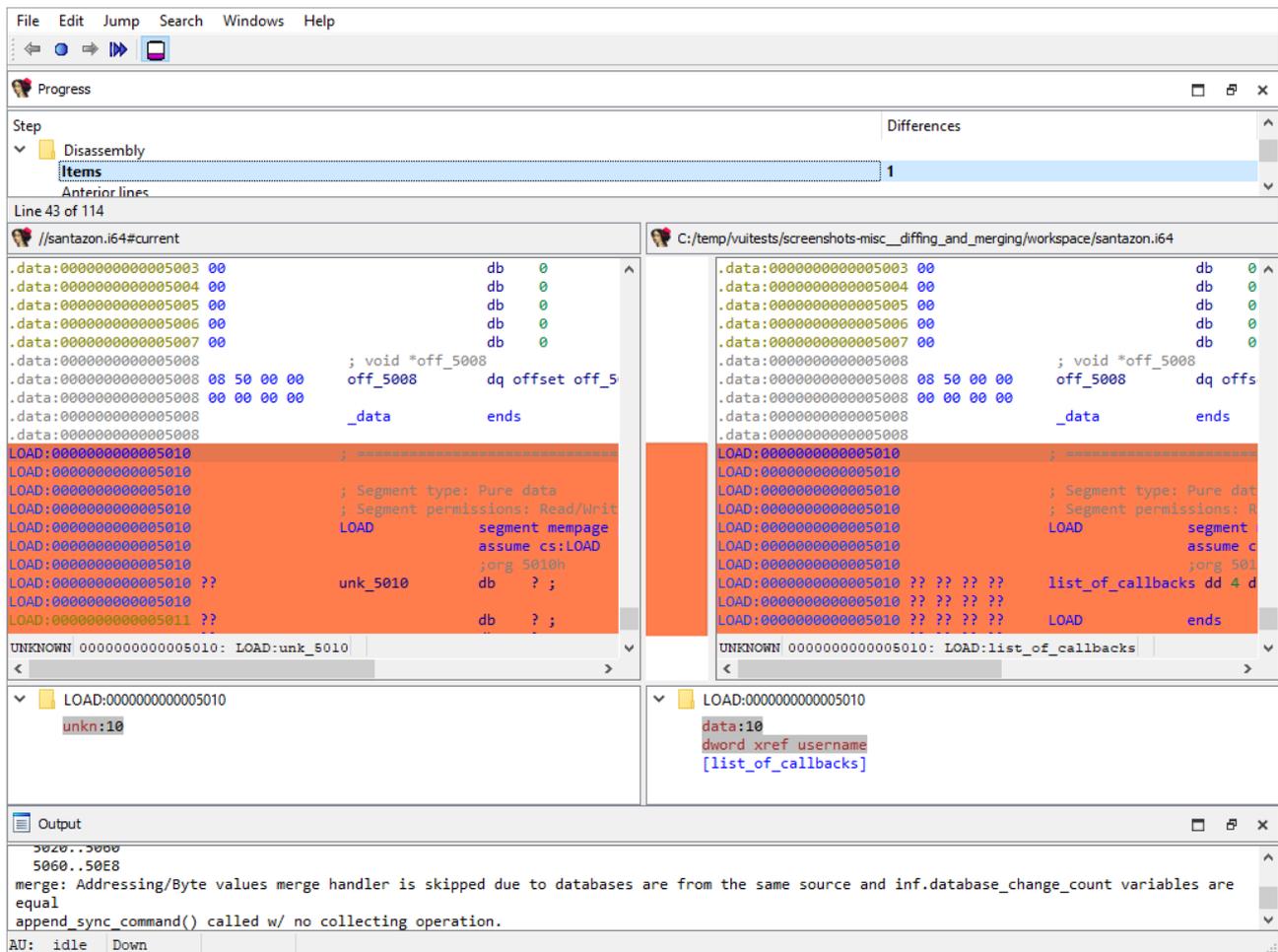
Before continuing, you might want to take a quick look at the tutorial for HVUI, the GUI client for IDA Teams' revision control functionality. It will be referenced multiple times in this document, although here we will focus specifically on the merging functionality.

2. Inspecting changes

After having done some reverse-engineering work on an IDA database, it is possible to view those changes in a special mode in IDA: right-click, and choose the diff action:



Here a new instance of IDA will be launched in a special "diff" mode:



2.1. IDA's diff mode

This new IDA mode lets the user compare two databases, in a traditional "diff" fashion: essentially a two-panel window, showing the unmodified file on the left and the version with your changes on the right.

2.1.1. The "Progress" widget



Represents the current [step](#) in the diff process.

2.1.2. The left panel

```

//santazon.i64#current
.data:0000000000005003 00 db 0
.data:0000000000005004 00 db 0
.data:0000000000005005 00 db 0
.data:0000000000005006 00 db 0
.data:0000000000005007 00 db 0
.data:0000000000005008 ; void *off_5008
.data:0000000000005008 08 50 00 00 off_5008 dq offset off_5
.data:0000000000005008 00 00 00 00
.data:0000000000005008 _data ends
.data:0000000000005008
LOAD:0000000000005010 ; =====
LOAD:0000000000005010
LOAD:0000000000005010 ; Segment type: Pure data
LOAD:0000000000005010 ; Segment permissions: Read/Writ
LOAD:0000000000005010 LOAD segment mempage
LOAD:0000000000005010 assume cs:LOAD
LOAD:0000000000005010 ;org 5010h
LOAD:0000000000005010 ?? unk_5010 db ? ;
LOAD:0000000000005010
LOAD:0000000000005011 ?? db ? ;
UNKNOWN| 0000000000005010: LOAD:unk_5010
LOAD:0000000000005010
  unkn:10

```

Shows the "untouched" version of the database (i.e., the one without your changes)

2.1.3. The right panel

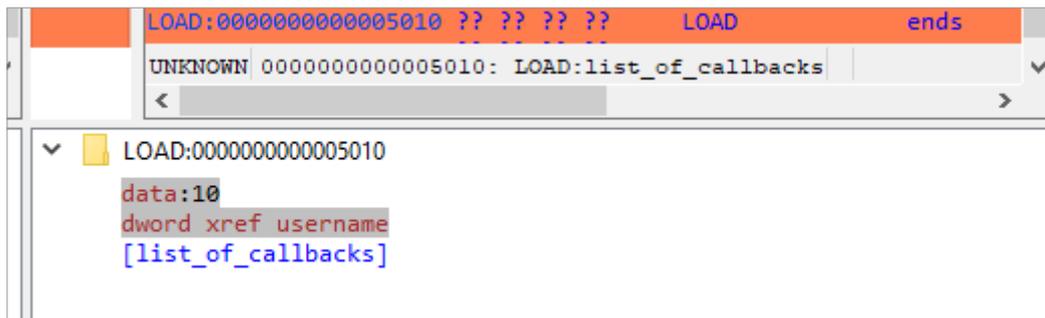
```

C:/temp/vuitests/screenshots-misc__diffing_and_merging/workspace/santazon.i64
.data:0000000000005003 00 db 0 ^
.data:0000000000005004 00 db 0
.data:0000000000005005 00 db 0
.data:0000000000005006 00 db 0
.data:0000000000005007 00 db 0
.data:0000000000005008 ; void *off_5008
.data:0000000000005008 08 50 00 00 off_5008 dq offs
.data:0000000000005008 00 00 00 00
.data:0000000000005008 _data ends
.data:0000000000005008
LOAD:0000000000005010 ; =====
LOAD:0000000000005010
LOAD:0000000000005010 ; Segment type: Pure dat
LOAD:0000000000005010 ; Segment permissions: R
LOAD:0000000000005010 LOAD segment
LOAD:0000000000005010 assume c
LOAD:0000000000005010 ;org 501
LOAD:0000000000005010 ?? ?? ?? ?? list_of_callbacks dd 4 d
LOAD:0000000000005010 ?? ?? ?? ??
LOAD:0000000000005010 ?? ?? ?? ?? LOAD ends
UNKNOWN| 0000000000005010: LOAD:list_of_callbacks
LOAD:0000000000005010
data:10
dword xref username
[list_of_callbacks]

```

Shows your version of the database (i.e., featuring your changes)

2.1.4. Diff region details



Notice how both panels have a little area at the bottom, that is labeled "Details".

Details are available on certain steps of the diffing process, and provide additional information about the change that is currently displayed.

2.1.5. The "diffing" toolbar



The actions in the toolbar are:

- Previous chunk
- Center chunk
- Next chunk
- Proceed to the next step
- Toggle 'Details'

Using actions in the toolbar, you can now iterate through the differences between the two databases, with each change shown in context as if viewed through a normal IDA window.

The ability to view changes in context was a major factor in the decision to use IDA itself as the diffing/merging tool for IDA Teams.

Diff mode IDA's toolbar actions

Previous chunk

Move to the previous change

Center chunk

Re-center the panels to show the current chunk (useful if you navigated around to get more context)

Next chunk

Move to the next change

Proceed to the next step

Move to the next [step](#) in the diffing process.

Toggle 'Details'

Toggle the visibility of the "Details" widgets in the various panels (note that some steps do not provide details, so even if the "Details" are requested, they might not be currently visible.)

2.2. Terminology

It is important to note the difference between the terms "diff" and "merge".

This document will sometimes use the two terms interchangeably. This is because to IDA, a diff is just a specialized merge. Both diffing and merging are handled by IDA's "merge mode", which involves up to 3 databases, one of which can be modified to contain the result of the merge.

A diff is simply a merge operation that involves only 2 databases, neither of which are modified.

This is why often times you will see the term "merge" used in the context of a diff. In this case "merge" is referring to IDA's "merge mode", rather than the process of merging multiple databases together into a combined database.

2.3. Using IDA as a diffing tool

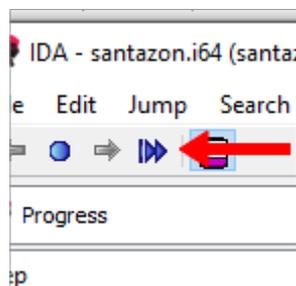
We must stress the fact that performing a merge between two IDA databases is quite different than performing a merge between, say, two text files. A change in a chunk of text file will not have an impact over another chunk.

IDA databases are not so simple. A change in one place in an idb will often have an impact on another place. For example, if a structure `mystruct` changed between two databases, it will have an impact not only on the name of the structure, but on cross-references to structure members, function prototypes, etc.

This is why IDA's merge mode is split into a strict series of "steps":



Within a single step it is possible to go forward & backward between different chunks. But because of possible inter-dependencies between steps, it is not possible to move backwards between steps, you can only go forward:



Since IDA's diff mode is just a variation of its merge mode, diffing databases is also subject to this sequential application of steps in order to view certain bits of information. That is why, in some steps (e.g., the "Disassembly/Items") IDA might not report some changes that were performed at another level.

For instance, if a user marked a function as `noret`, the listings that will be shown in "Disassembly/Items" step, will not advertise that there was a change at that place (even though the "`Attributes: noreturn`" is visible in the left-hand listing), only the changes to the instructions (and data, ...) are visible in the current step:

File Edit Jump Search Windows Help

Progress

Step: Anterior lines: 0, Posterior lines: 0, **EA additional flags: 1**

Line 46 of 114

```

//santazon.i64#current
.text:00000000000011F0 ; ===== S U B R O U T
.text:00000000000011F0
.text:00000000000011F0
.text:00000000000011F0
.text:00000000000011F0 000 48 8D 3D 19 ...rrrr dump_core proc near
.text:00000000000011F0 000 3E 00 00 lea rdi, un
.text:00000000000011F7 000 48 8D 05 12 ...rrrr lea rax, un
.text:00000000000011F7 000 3E 00 00
.text:00000000000011FE 000 48 39 F8 cmp rax, rd
.text:0000000000001201 000 74 15 jz short l
.text:0000000000001203 000 48 8B 05 CE ...rrrr mov rax, cs
.text:0000000000001203 000 3D 00 00
.text:000000000000120A 000 48 85 C0 test rax, ra
.text:000000000000120D 000 74 09 jz short l
.text:000000000000120F 000 FF E0 jmp rax
.text:000000000000120F ; -----
.text:0000000000001211 000 0F 1F 80 00... align 8
.text:0000000000001218 locret_1218:
.text:0000000000001218
.text:0000000000001218 000 C3 retn
000011F0 00000000000011F0: dump_core

C:/temp/vuitests/screenshots-misc_diffing_caveat/workspace/santazon.i64
.text:00000000000011F0 ; ===== S U B
.text:00000000000011F0
.text:00000000000011F0 ; Attributes: noreturn
.text:00000000000011F0
.text:00000000000011F0 dump_core proc ne
.text:00000000000011F0 000 48 8D 3D 19 ...rrrr lea
.text:00000000000011F0 000 3E 00 00
.text:00000000000011F7 000 48 8D 05 12 ...rrrr lea
.text:00000000000011F7 000 3E 00 00
.text:00000000000011FE 000 48 39 F8 cmp
.text:0000000000001201 000 74 15 jz
.text:0000000000001203 000 48 8B 05 CE ...rrrr mov
.text:0000000000001203 000 3D 00 00
.text:000000000000120A 000 48 85 C0 test
.text:000000000000120D 000 74 09 jz
.text:000000000000120F 000 FF E0 jmp
.text:000000000000120F ; -----
.text:0000000000001211 000 0F 1F 80 00... align 8
.text:0000000000001218 locret_1218:
.text:0000000000001218
.text:0000000000001218
000011F0 00000000000011F0: dump_core
    
```

Output

```

5020..5000
5060..50E8
merge: Addressing/Byte values merge handler is skipped due to databases are from the same source and inf.database_change_count variables are equal
append_sync_command() called w/ no collecting operation.
AU: idle Down
    
```

The change will, however, be visible at a later step (i.e., "Functions/Registry"):

File Edit Jump Search Windows Help

Progress

Step: Source file ranges: 0, **Registry: 1**

Line 51 of 114

Function name	R	F	L	M	O	S	B	T	=
f _realloc	R	T	.
f _atoi	R	T	.
f _isoc99_scanf	R	T	.
f start	T	.
f dump_core	R
f sub_1260	R
f print_welcome_screen	R	B	T	.
f register_instance	R	B	T	.
f load_dylib	R	B	T	.
f marshall	R	B	T	.

Line 29 of 56

```

dump_core
start_ea: 11F0
end_ea : 1219
Attributes: sp-ready prolog-analysis-ok purged-analysis-ok
lvars-size: 0
saved-regs: 0
npurged : 0

dump_core
start_ea: 11F0
end_ea : 1219
Attributes: noreturn sp-ready prolog-analysis-ok purged-analysis-ok
lvars-size: 0
saved-regs: 0
npurged : 0
    
```

Output

```

Caching '//santazon.i64#current:Functions'... ok
Caching 'C:/temp/vuitests/screenshots-misc_diffing_caveat/workspace/santazon.i64:Functions'... ok
Caching '//santazon.i64#current:Functions'... ok
Caching 'C:/temp/vuitests/screenshots-misc_diffing_caveat/workspace/santazon.i64:Functions'... ok
AU: idle Down
    
```

NOTE

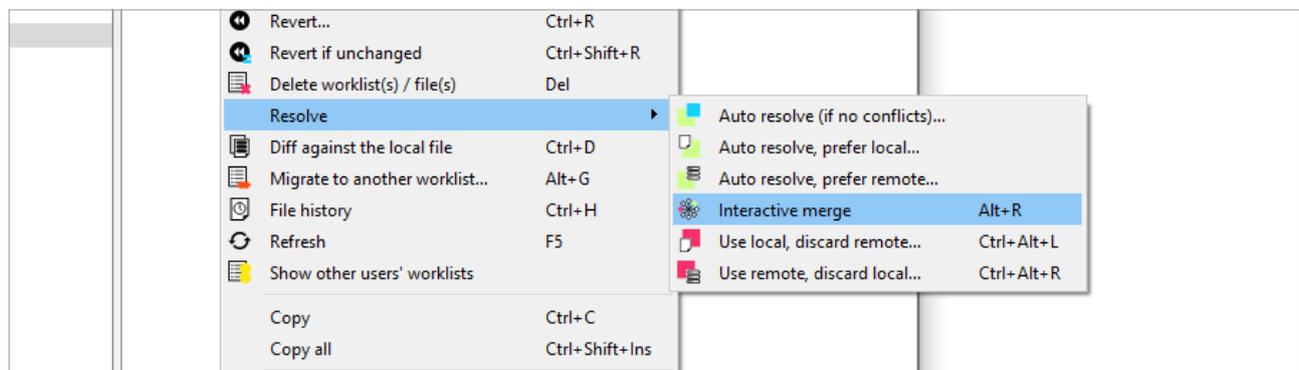
The changes applied during the "diff" process are only temporary. Exiting IDA (at any moment) will not alter the files being compared.

2.4. Merging concurrent modifications (conflicts)

As with any collaborative tool, it may happen that two coworkers work on the same dataset (e.g., IDA database), and make modifications to the same areas, resulting in "conflicts". Conflicts must be "resolved" prior to committing.

Name	Last changed	Size
1, joe@joe_on_joesbox	2023-05-24 15:02:19	1
✓ //santazon.i64#1/2	2023-05-24 15:02:19	54.6k

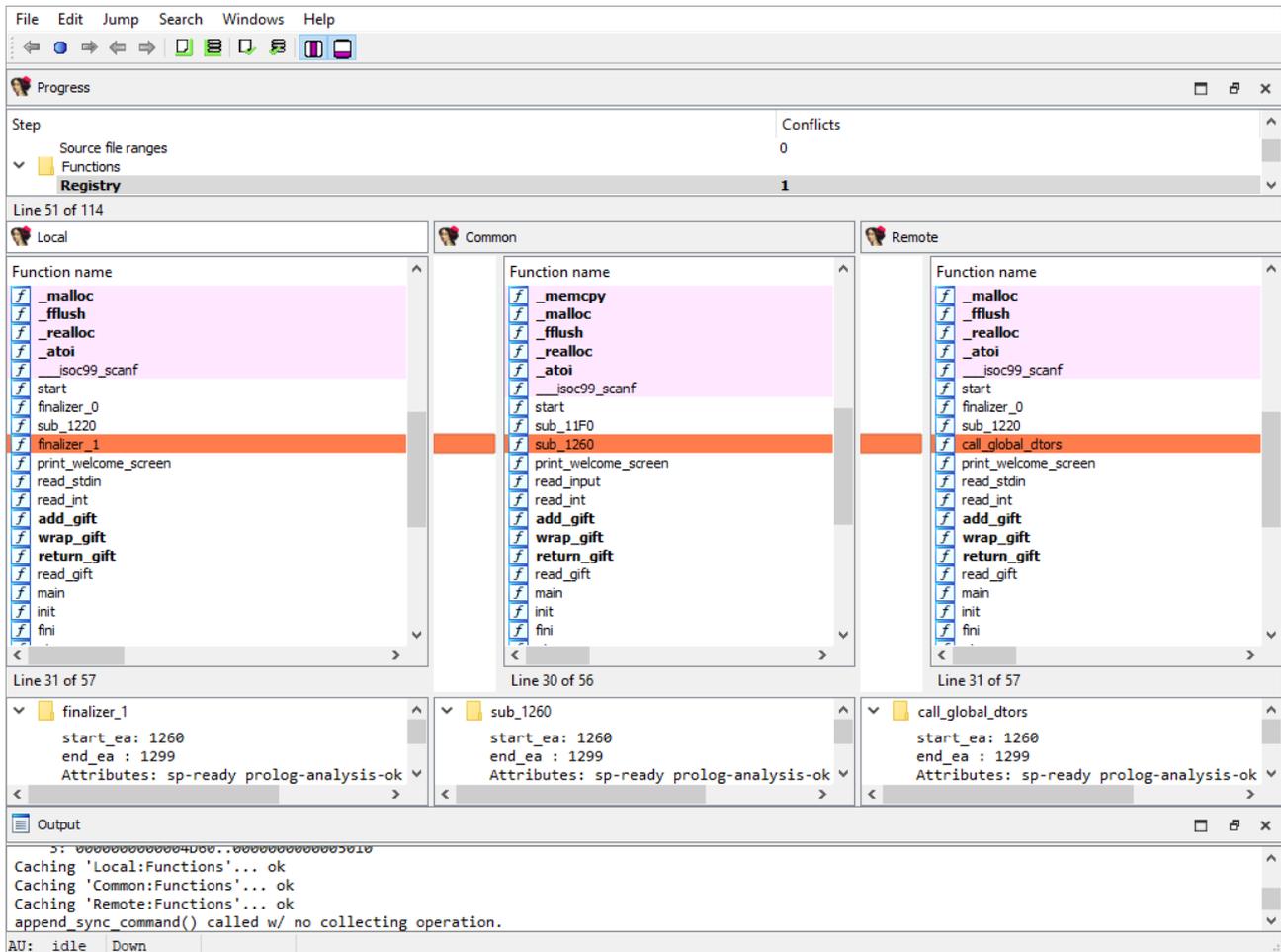
To do that, right-click and pick one of the "resolve" options:



IDA Teams provides the following merge strategies.

2.4.1. Interactive merging

If the option that was chosen (e.g., [Interactive merge mode](#)) requires user interaction due to conflicts, IDA will show in 3-pane "merge" mode.



When a conflict is encountered, you'll have the ability to pick, for all conflicts, which change should be kept (yours, or the other). Every time you pick a change (and thus resolve a conflict), IDA will proceed with the merging, applying all the non-conflicting changes it can, until the next conflict - if any. When all conflicts are resolved, you can leave IDA, and the new resulting file is ready to be submitted.

3. Appendix A

3.1. Merge Steps

This section provides a detailed overview of the steps involved in the merge process. The list of predefined merge steps is defined in `merge.hpp` of the IDASDK:

```
enum merge_kind_t
{
    MERGE_KIND_NETNODE,           ///< netnode (no merging, to be used in idbunits)
    MERGE_KIND_AUTOQ,            ///< auto queues
    MERGE_KIND_INF,              ///< merge the inf variable (global settings)
    MERGE_KIND_ENCODINGS,       ///< merge encodings
    MERGE_KIND_ENCODINGS2,      ///< merge default encodings
    MERGE_KIND_SCRIPTS2,        ///< merge scripts common info
    MERGE_KIND_SCRIPTS,         ///< merge scripts
    MERGE_KIND_CUSTDATA,        ///< merge custom data type and formats
    MERGE_KIND_STRUCTS,         ///< merge structs (globally: add/delete structs entirely)
    MERGE_KIND_STRMEM,          ///< merge struct members
    MERGE_KIND_ENUMS,           ///< merge enums
    MERGE_KIND_TILS,            ///< merge type libraries
    MERGE_KIND_TINFO,           ///< merge tinfo
    MERGE_KIND_UDTMEM,          ///< merge UDT members (local types)
    MERGE_KIND_SELECTORS,       ///< merge selectors
    MERGE_KIND_STT,             ///< merge flag storage types
    MERGE_KIND_SEGMENTS,        ///< merge segments
    MERGE_KIND_SEGGRRPS,        ///< merge segment groups
    MERGE_KIND_SEGREGS,         ///< merge segment registers
}
```

```

MERGE_KIND_ORPHANS,        ///< merge orphan bytes
MERGE_KIND_BYTEVAL,       ///< merge byte values
MERGE_KIND_FIXUPS,        ///< merge fixups
MERGE_KIND_MAPPING,       ///< merge manual memory mapping
MERGE_KIND_EXPORTS,       ///< merge exports
MERGE_KIND_IMPORTS,       ///< merge imports
MERGE_KIND_PATCHES,       ///< merge patched bytes
MERGE_KIND_FLAGS,         ///< merge flags_t
MERGE_KIND_EXTRACT,       ///< merge extra next or prev lines
MERGE_KIND_AFLAGS_EA,     ///< merge aflags for mapped EA
MERGE_KIND_IGNOREMICRO,   ///< IM ("I ignore micro") flags
MERGE_KIND_HIDDENRANGES,  ///< merge hidden ranges
MERGE_KIND_SOURCEFILES,   ///< merge source files ranges
MERGE_KIND_FUNC,          ///< merge func info
MERGE_KIND_FRAMEMGR,      ///< merge frames (globally: add/delete frames entirely)
MERGE_KIND_FRAME,         ///< merge function frame info (frame members)
MERGE_KIND_STKPNTS,       ///< merge SP change points
MERGE_KIND_FLOWS,         ///< merge flows
MERGE_KIND_CREFS,         ///< merge crefs
MERGE_KIND_DREFS,         ///< merge drefs
MERGE_KIND_BPTS,          ///< merge breakpoints
MERGE_KIND_WATCHPOINTS,   ///< merge watchpoints
MERGE_KIND_BOOKMARKS,     ///< merge bookmarks
MERGE_KIND_TRYBLKS,       ///< merge try blocks
MERGE_KIND_DIRTREE,       ///< merge std dirtrees
MERGE_KIND_VFTABLES,     ///< merge vftables
MERGE_KIND_SIGNATURES,    ///< signatures
MERGE_KIND_PROBLEMS,      ///< problems
MERGE_KIND_UI,            ///< UI
MERGE_KIND_NOTEPAD,       ///< notepad
MERGE_KIND_LOADER,        ///< loader data
MERGE_KIND_DEBUGGER,      ///< debugger data
MERGE_KIND_LAST,          ///< last predefined merge handler type.
                           ///< please note that there can be more merge handler types,
                           ///< registered by plugins and processor modules.
};

```

The list of merge steps is not final. If for example there is a conflict in structure members then the new merge phase to resolve this conflict will be created. The same is hold for UDT, functions, frames and so on. In other words in general case the exact number of merge steps is undefined and depends on the databases.

Each item in a merge step is assigned to a difference position named `diffpos`. It may be an EA (effective address), enum id, structure member offset, artificial index and so on. In other words, a `diffpos` is a way of addressing something in the database.

Every merge step starts with the calculation of differences and conflicts between items at the corresponding difference positions. As the result there is a list of `diffpos` with differences or conflicts. The `diffpos`'s without differences are not included in the list. Adjacent `diffpos`'s are combined into a difference range called `diffrange`.

The merging process operates on a difference range `diffrange`. For one `diffrange`, a single merge policy can be selected.

3.1.1. Global settings/Database attributes

Merging of global database attributes. These attributes are mainly stored in the `idainfo` structure. This phase has two subphases:

- Global settings/Database attributes/Graph mode
- Global settings/Database attributes/Text mode

The "Detail" pane is absent.

File Edit Jump Search Windows Help

Progress

Step Global settings Database attributes 2 Conflicts

Line 2 of 111

Local

Attribute	Value
listing.show_hidden_funcs	false
listing.show_hidden_insns	false
listing.show_hidden_segms	false
listing.show_src_linnum	false
listing.use_graph_view	false
listing.use_unsupported_asm_directives	true
listing.xref_margin	0x50
names.dummy_names	ea
names.list.include_autogenerated_names	true
names.list.include_public_names	true
names.list.include_regular_names	true
names.list.include_weak_names	false
names.max_autogenerated_name_length	0xF
pc.decode_fpu_insns	true
special_segment_entry_size	4
strits.break	0xC
strits.display_strlit_xrefs	true
strits.generate_strlit_names	true
strits.layout	zero-terminated
strits.leading_zeroes	0
strits.name_prefix	a
strits.preserve_case	false
strits.serial_number	0
strits.set_autogenerated_bit	true
strits.unicode_seen	false
strits.unit_width	1 byte
strits.use_serial_names	false
suspiciousness_limits.high	0x8049748
suspiciousness_limits.low	0x8048000
target.assembler	0
target.processor	metapc
test_mode	false

Remote

Attribute	Value
listing.show_hidden_funcs	false
listing.show_hidden_insns	false
listing.show_hidden_segms	false
listing.show_src_linnum	false
listing.use_graph_view	false
listing.use_unsupported_asm_directives	true
listing.xref_margin	0x50
names.dummy_names	ea
names.list.include_autogenerated_names	true
names.list.include_public_names	true
names.list.include_regular_names	true
names.list.include_weak_names	false
names.max_autogenerated_name_length	0xF
pc.decode_fpu_insns	true
special_segment_entry_size	4
strits.break	0xB
strits.display_strlit_xrefs	true
strits.generate_strlit_names	true
strits.layout	zero-terminated
strits.leading_zeroes	0
strits.name_prefix	a
strits.preserve_case	false
strits.serial_number	0
strits.set_autogenerated_bit	true
strits.unicode_seen	false
strits.unit_width	1 byte
strits.use_serial_names	false
suspiciousness_limits.high	0x8049748
suspiciousness_limits.low	0x8048000
target.assembler	0
target.processor	metapc
test_mode	false

Line 139 of 162

3.1.2. Global settings/Processor specific

Merging of global processor options. Usually these options are stored in the `idpflags` netnode.

The "Detail" pane is absent.

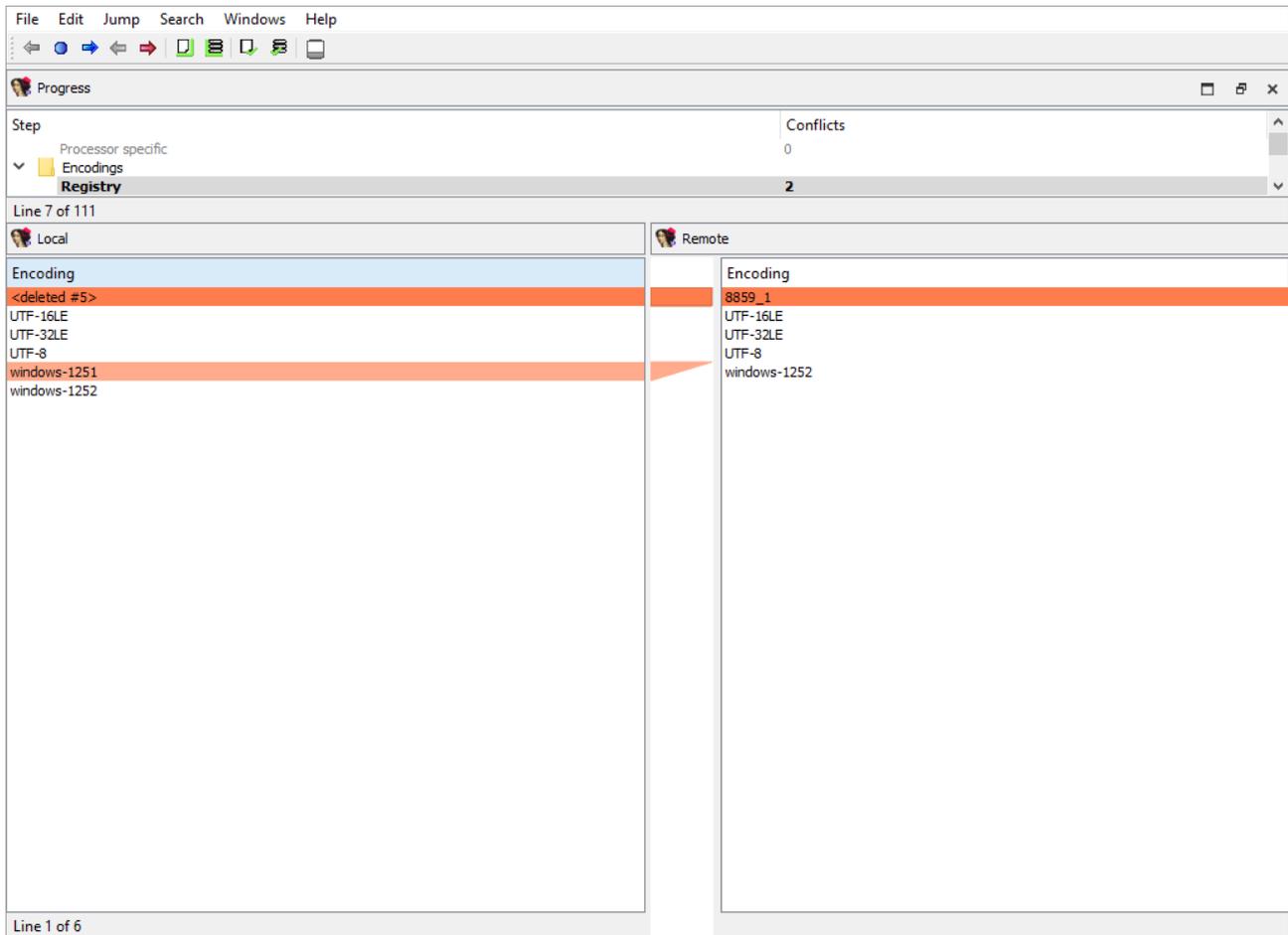
Step	Conflicts
Graph mode	0
Text mode	0
Processor specific	4

Local	Remote
Attribute	Attribute
arm.analysis.simplify_instructions: true	arm.analysis.simplify_instructions: true
arm.analysis.disable_pointer_dereferencing: false	arm.analysis.disable_pointer_dereferencing: false
arm.analysis.architecture_v5_or_higher: true	arm.analysis.architecture_v5_or_higher: true
arm.analysis.no_automatic_THUMB_switch: false	arm.analysis.no_automatic_THUMB_switch: false
arm.analysis.disable_BL_jumps_detection: true	arm.analysis.disable_BL_jumps_detection: true
arm.analysis.MOVT_pair_handling_options: Valid addresses only	arm.analysis.MOVT_pair_handling_options: Valid addresses only
arm.arch.base_architecture: ARM_meta	arm.arch.base_architecture: ARMv6
arm.arch.profile: unknown	arm.arch.profile: unknown
arm.arch.VFP_instructions: v8	arm.arch.VFP_instructions: v2
arm.arch.advanced_SIMD: v8	arm.arch.advanced_SIMD: none
arm.arch.ARM_instructions: Yes	arm.arch.ARM_instructions: Yes
arm.arch.thumb_instructions: Thumb-2	arm.arch.thumb_instructions: Thumb
arm.arch.XScale_architecture: Yes	arm.arch.XScale_architecture: No
arm.arch.wireless_MMX: WMMXv2	arm.arch.wireless_MMX: None
arm.arch.half_precision_extension: Yes	arm.arch.half_precision_extension: No
arm.arch.thumb2_EE_extension: Yes	arm.arch.thumb2_EE_extension: No
arm.arch.be8: No	arm.arch.be8: No
arm.arch.arm64_32: No	arm.arch.arm64_32: No
arm.arch_name: metaarm	arm.arch_name: ARMv6
arm.core_name: Generic metaarm core	arm.core_name: Generic ARMv6 core
arm.mach_header: CFFAEDFE0C000001000000000B000000020000000B000000001000000000000000	arm.mach_header:
arm.input_file_image_base: 0xFFFFFFFF081DD310	arm.input_file_image_base: 0
arm.input_file_uuid: 297D45D9A3CB43B1B5C114ED6637FC72	arm.input_file_uuid:

3.1.3. Encodings/Registry

Merging of registered string literal encodings. These encodings are used to properly display string literal in the disassembly listing.

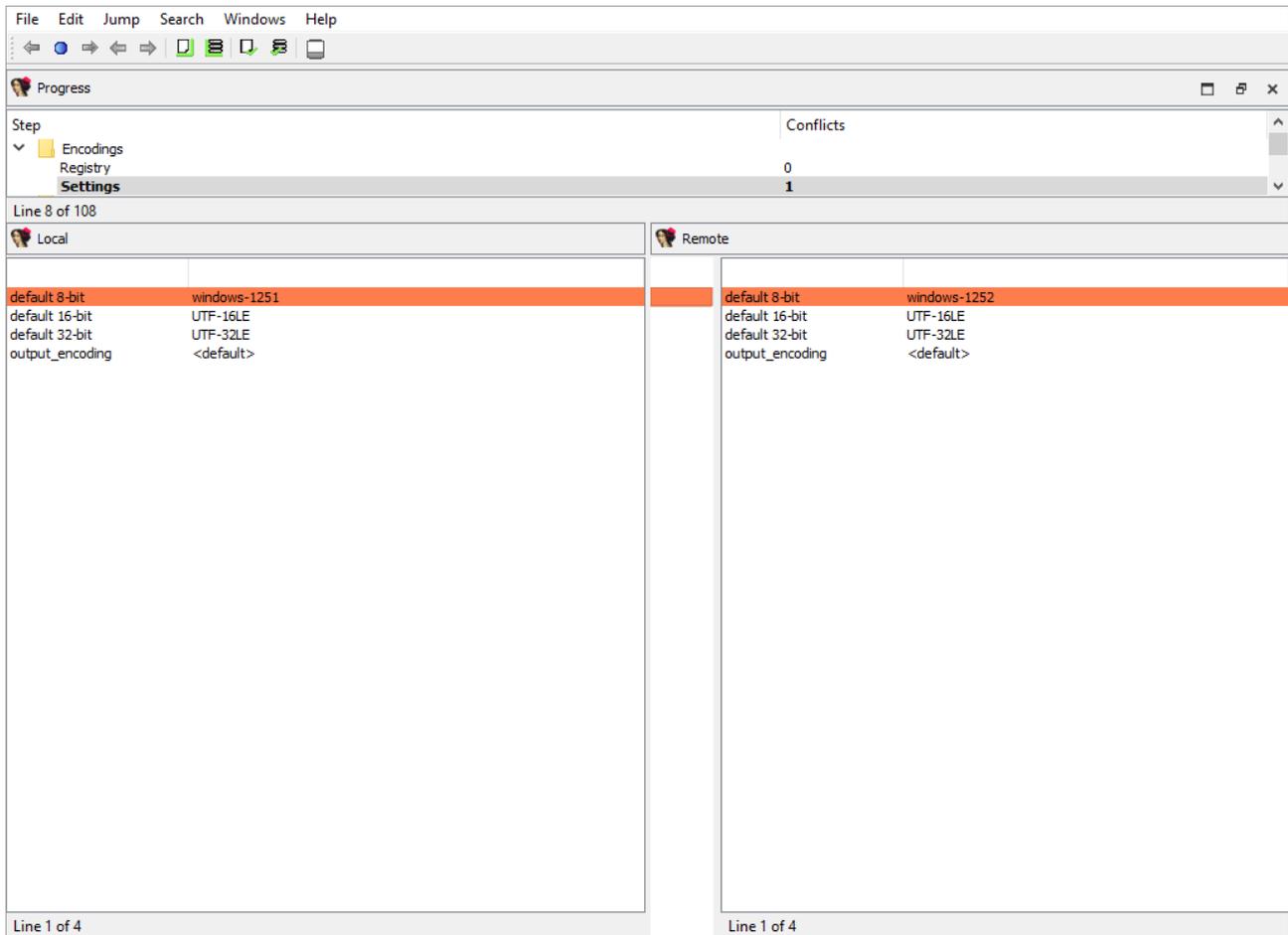
The "Detail" pane is absent.



3.1.4. Encodings/Settings

Merging of default string encodings: what string encoding among the registered ones are considered as the default ones.

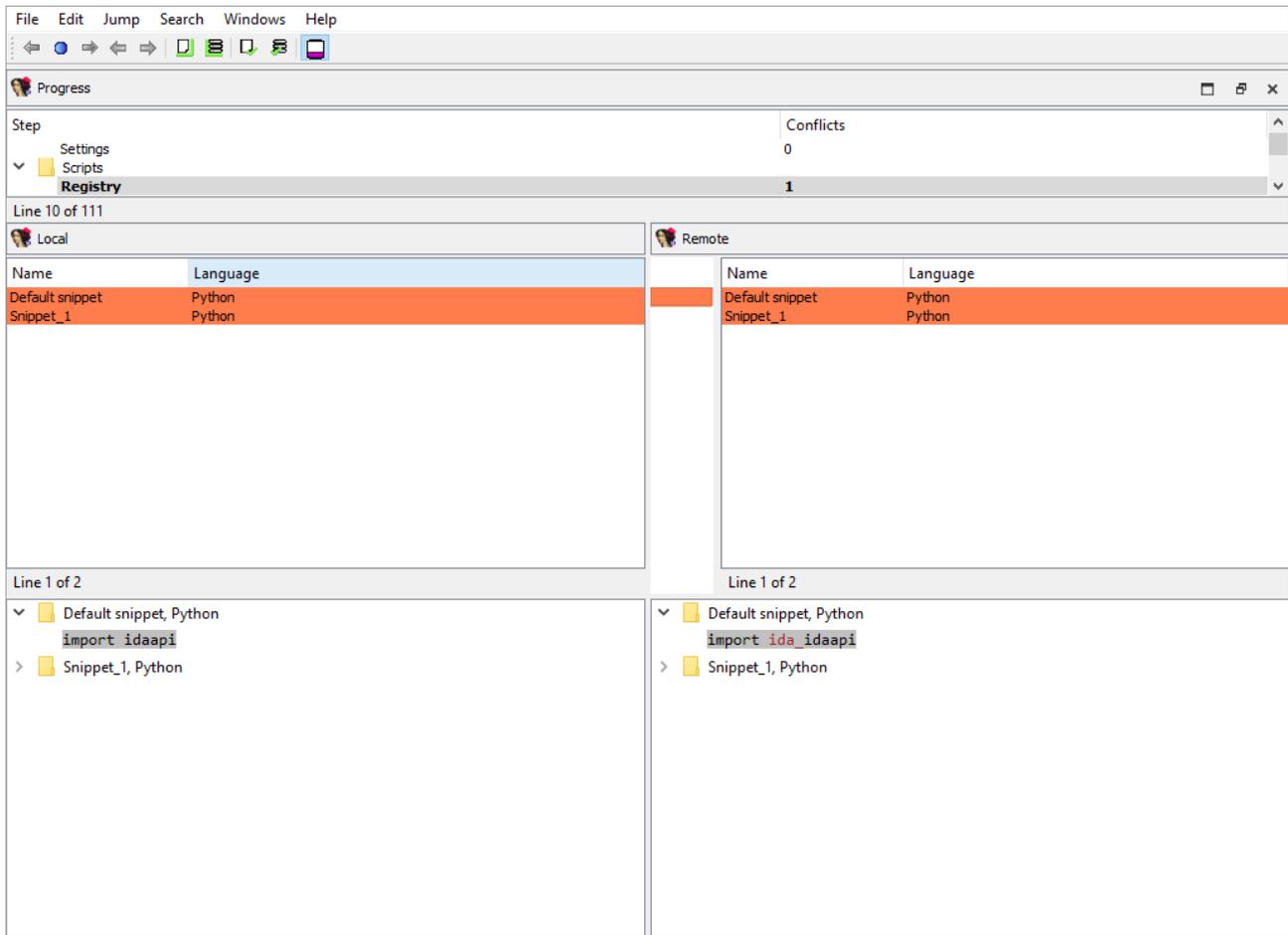
The "Detail" pane is absent.



3.1.5. Scripts/Registry

Merging of embedded script snippets.

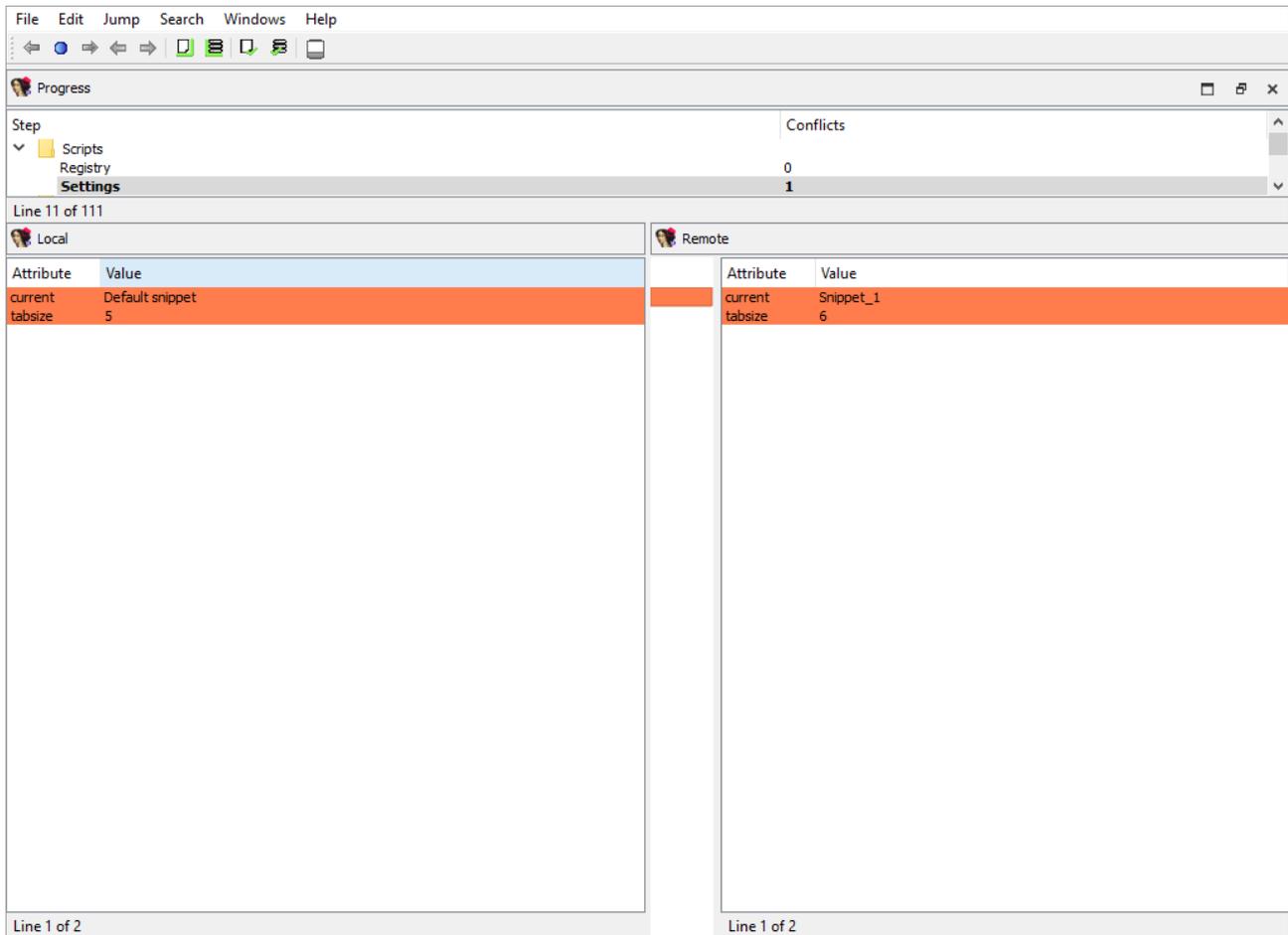
When merging of embedded script snippets, the script name/language is displayed, and the "Detail" pane contains the script source with the highlighted differences:



3.1.6. Scripts/Settings

Merging of the default snippet and tabulation size.

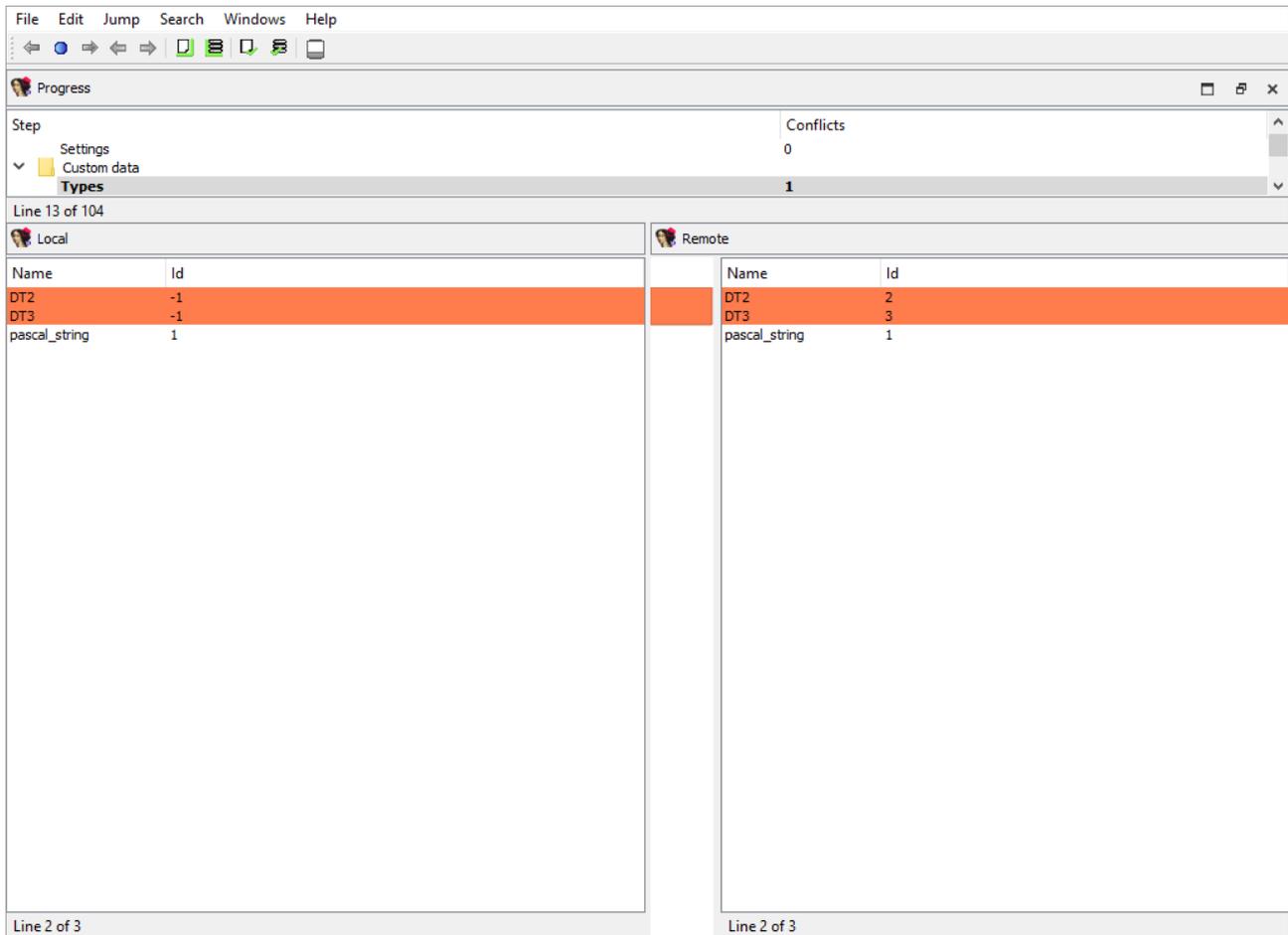
The "Detail" pane is absent.



3.2. Custom data/Types and Custom data/Formats

Merging of the registered custom data types and formats.

The "Detail" pane is absent.



3.2.1. Types/Enums

Merging of assembler level enums (`enum_t`). Ghost enums are skipped in this phase, they will be merged when handling local types.

To calculate `diffpos`, IDA Teams matches enum members by name and maps all enums with common member names into one `diffpos`.

An example of enum merging:

```

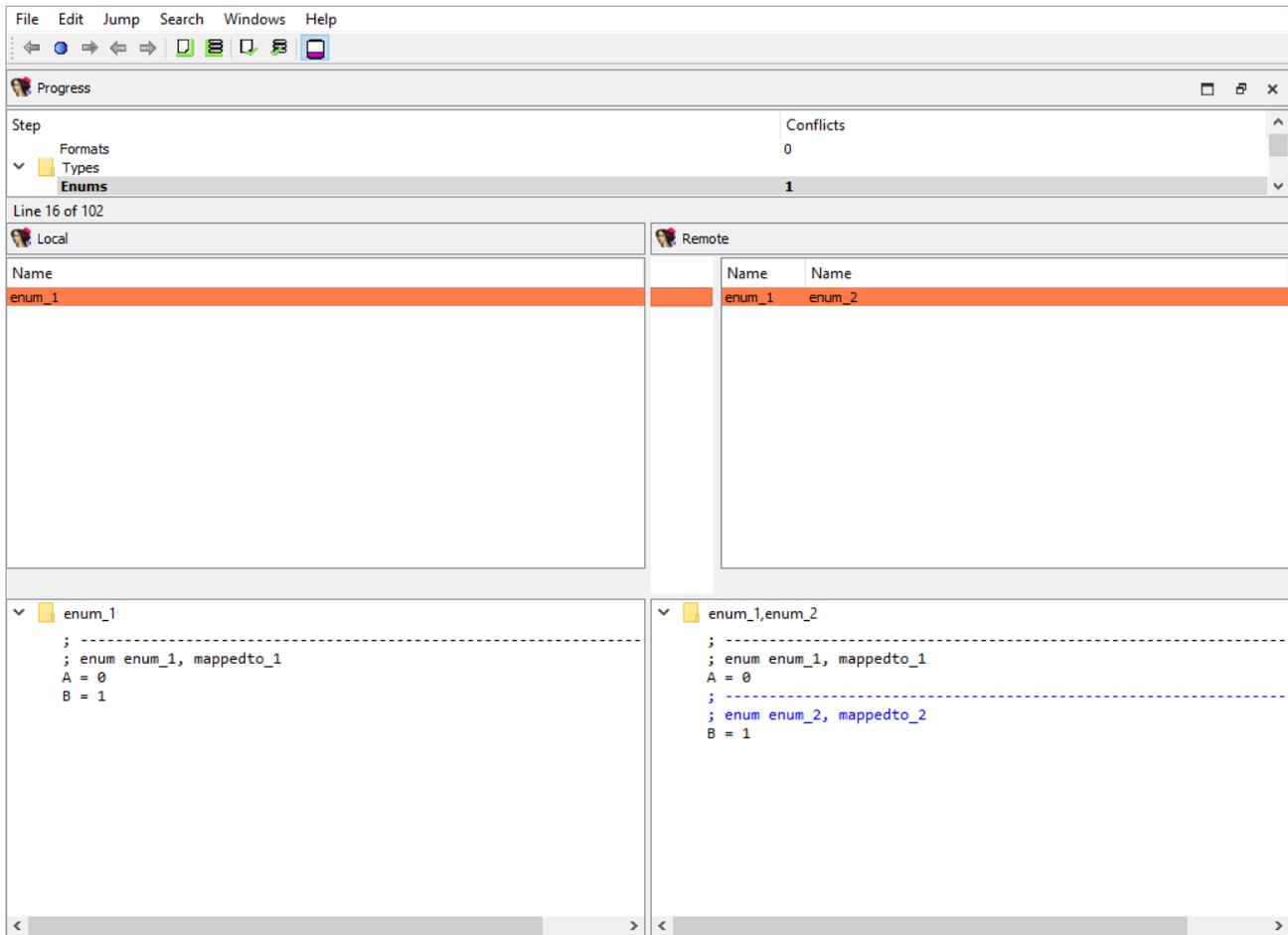
local_idb
;-----
; enum enum_1, mappedto_1
A           = 0
B           = 1

remote_idb
;-----
; enum enum_1, mappedto_1
A           = 0
;-----
; enum enum_2, mappedto_2
B           = 1

```

In both idbs, enum constant "B" is present. However, in the remote idb "B" has a different parent enum, "enum_2". Therefore enum_1 in the local idb corresponds to enum_1 and enum_2 in the remote idb. The user can select either enum_1 from the local idb or enum_1 and enum_2 from the remote idb.

In other words, IDA will display both enum_1 and enum_2 in the Remote pane, indicating that the difference between the Local and Remote databases corresponds to two separate enums, but they are treated as a single difference location. The "Detail" pane will display the full enum definitions, with the differences highlighted:



3.2.2. Types/Structs

Merging of assembler level structures (`struc_t`).

To calculate `diffpos`, IDA Teams matches structs by the following attributes, in this order:

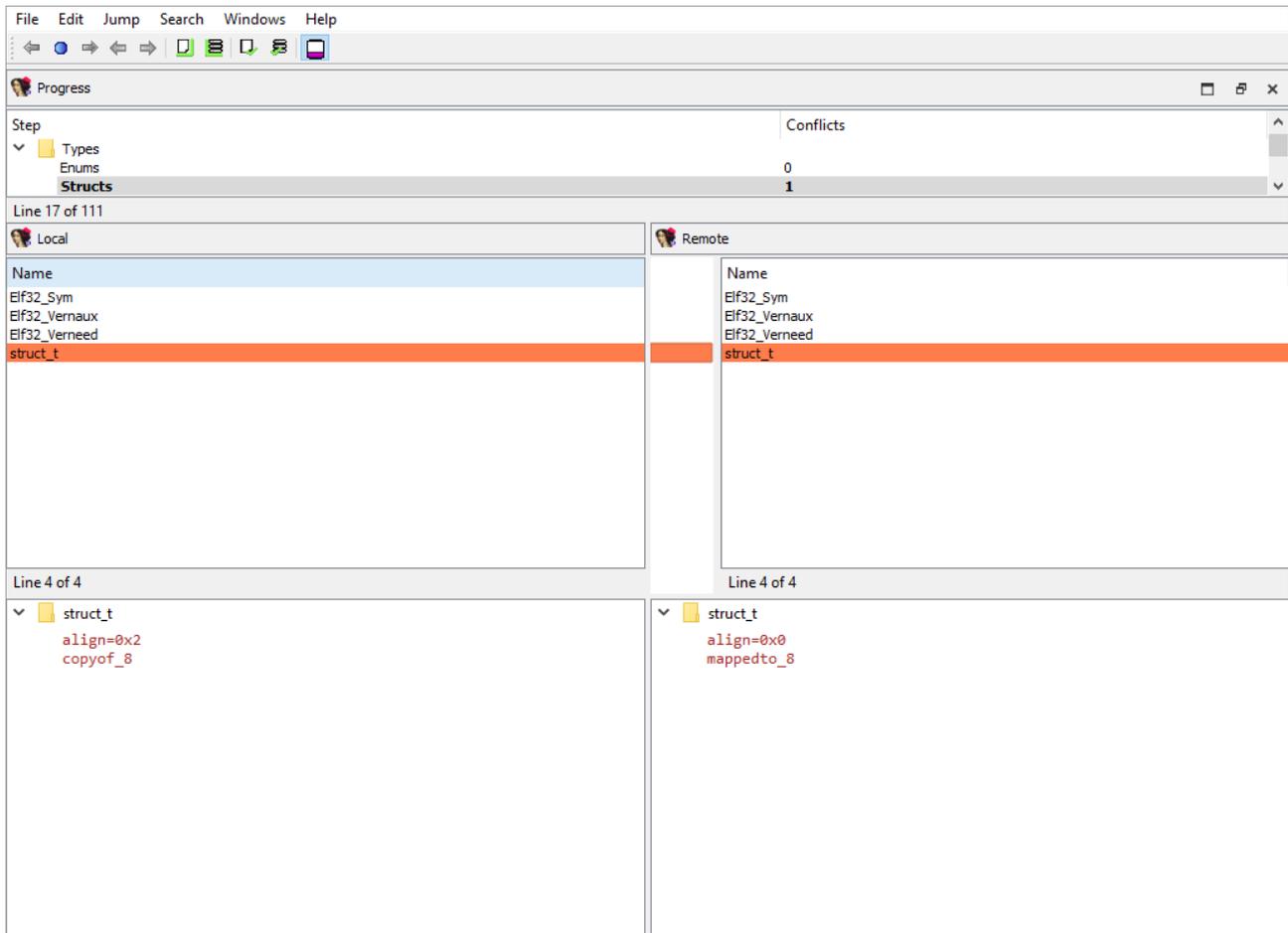
1. the structure name
2. the structure `tid` and size

If we fail to match a structure, then it will stay unmatched. Such an unmatched structure will have its own `diffpos`, allowing the user to copy it to the other idb or to delete it altogether.

This merge phase deals with the entire structure types and their attributes. Entire structure types may be added or deleted, and/or conflicts in the structure attributes are resolved.

If members of matched structures (at the same `diffpos`) differ, the conflict will be resolved later, during the **Types/Struct members/...** merge phase.

In the UI, IDA will display the list of structure names, with the "Detail" pane showing the structure attributes:

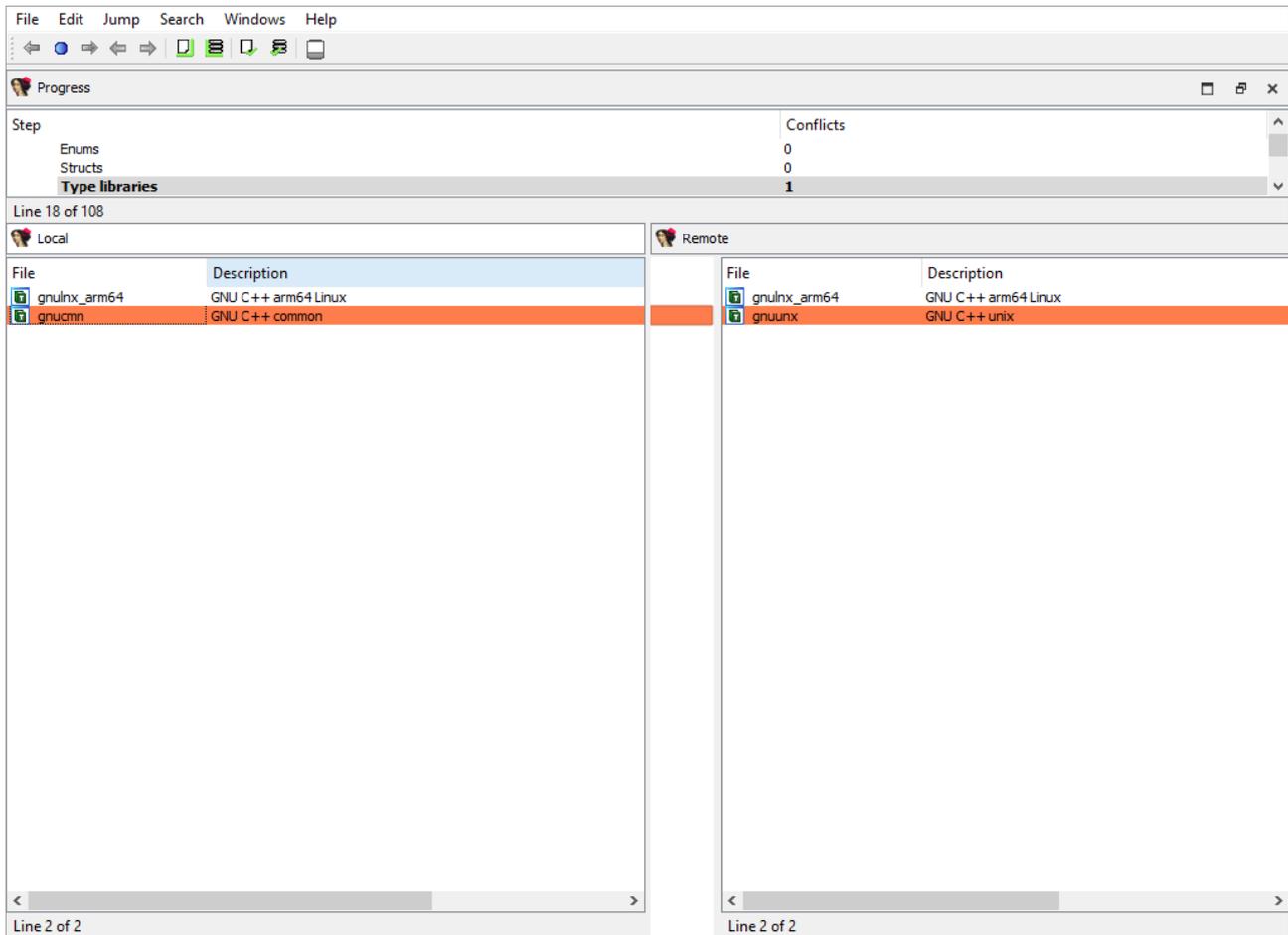


3.2.3. Types/Type libraries

Merging of the loaded type libraries.

This merge phase uses the standard "Type libraries" widget.

The "Detail" pane is absent.



3.2.4. Types/Local types

Merging of local types.

To calculate `diffpos`, IDA Teams matches local types by the following attributes, in this order:

1. the type name
2. the ordinal number and base type

If we fail to match a type, then it will stay unmatched. Such an unmatched type will have its own `diffpos`, allowing the user to copy it to the other idb or to delete it altogether.

This merge phase deals with entire types and their attributes. Entire local types may be added or deleted, and/or conflicts in their attributes are resolved. Differences in type members (e.g., struct members) will be resolved in a separate phase: **Types/Local type members**

This merge phase uses the standard "Local types" widget. The "Detail" pane displays the type definition and its attributes.

The screenshot shows a debugger interface with a 'Local types' window. At the top, a 'Progress' bar shows 'Step' with 'Structs' and 'Type libraries' counts of 0, and 'Local types' with a count of 1. Below this, there are two panes: 'Local' and 'Remote'. The 'Local' pane contains a table with the following data:

Ordinal	Name	Size	Sync	Description
1	main::\$8B67069E32DAC324...	00000004	Auto	struct { __int32 foo : 16;}
2	main::\$C9EFC2639E8A0EE0...	00000004	Auto	struct { __int32 foo : 32;}

The 'Remote' pane is currently empty. Below the tables, the 'Detail' pane shows the definition for the selected member (Ordinal 1):

```

main::$8B67069E32DAC32471CDECAC25F6845C
  struct main::$8B67069E32DAC32471CDECAC25F6845C { __int32 foo : 16; }
  COPYTO
  > main::$C9EFC2639E8A0EE0E99DC3CC9FC88CFA
  
```

3.2.5. Types/Struct members/... and Types/Local type members/...

For example:

- Types/Struct members/struct_t
- Types/Local type members/struct conflict_t

These merge phases merges the conflicting members of a structure or a local type.

The "Detail" pane displays full information about the current member along with its attributes.

The screenshot shows a debugger's merge phase for a struct type. The interface is divided into several sections:

- Progress:** Shows the current step as 'struct_t' with 'Local type members' and 'struct conflict_t' listed. The 'Conflicts' column shows 0 for 'struct_t' and 1 for 'struct conflict_t'.
- Local View:** Displays a table of struct members for 'struct conflict_t' at 'Line 23 of 116'.

Range	Member
0..20	f1
20..40	
- Remote View:** Displays a table of struct members for 'struct conflict_t' at 'Line 1 of 2'.

Range	Member
0..20	f1
20..40	f2
- Local Definition:** Shows the local struct definition at 'Line 1 of 2':


```

0..20
  int f1, bitoff:0x0
  effective alignment in bytes (effalign): 4
20..40
      
```
- Remote Definition:** Shows the remote struct definition at 'Line 1 of 2':


```

0..20
  char f1, bitoff:0x0
  effective alignment in bytes (effalign): 1
20..40
      
```

3.2.6. Types/Ghost struct comments

Ghost structs may have comments attached to them.

This merge phase handles these comments:

The screenshot displays a merge conflict resolution interface. At the top, a menu bar includes 'File', 'Edit', 'Jump', 'Search', 'Windows', and 'Help'. Below it is a 'Progress' window showing the current step: 'Type libraries', 'Local types', and 'Ghost struct comments'. A 'Conflicts' table is visible, with the following data:

Step	Conflicts
Type libraries	0
Local types	0
Ghost struct comments	1

The main area is split into four panes. The top-left pane, labeled 'Local', shows a list of identifiers with 'IID' highlighted in orange. The top-right pane, labeled 'Remote', shows the same list with 'IID' also highlighted in orange. The bottom-left pane shows the local comment for 'IID':

```
cmt: local comment
rptcmt: local repeatable comment
```

The bottom-right pane shows the remote comment for 'IID':

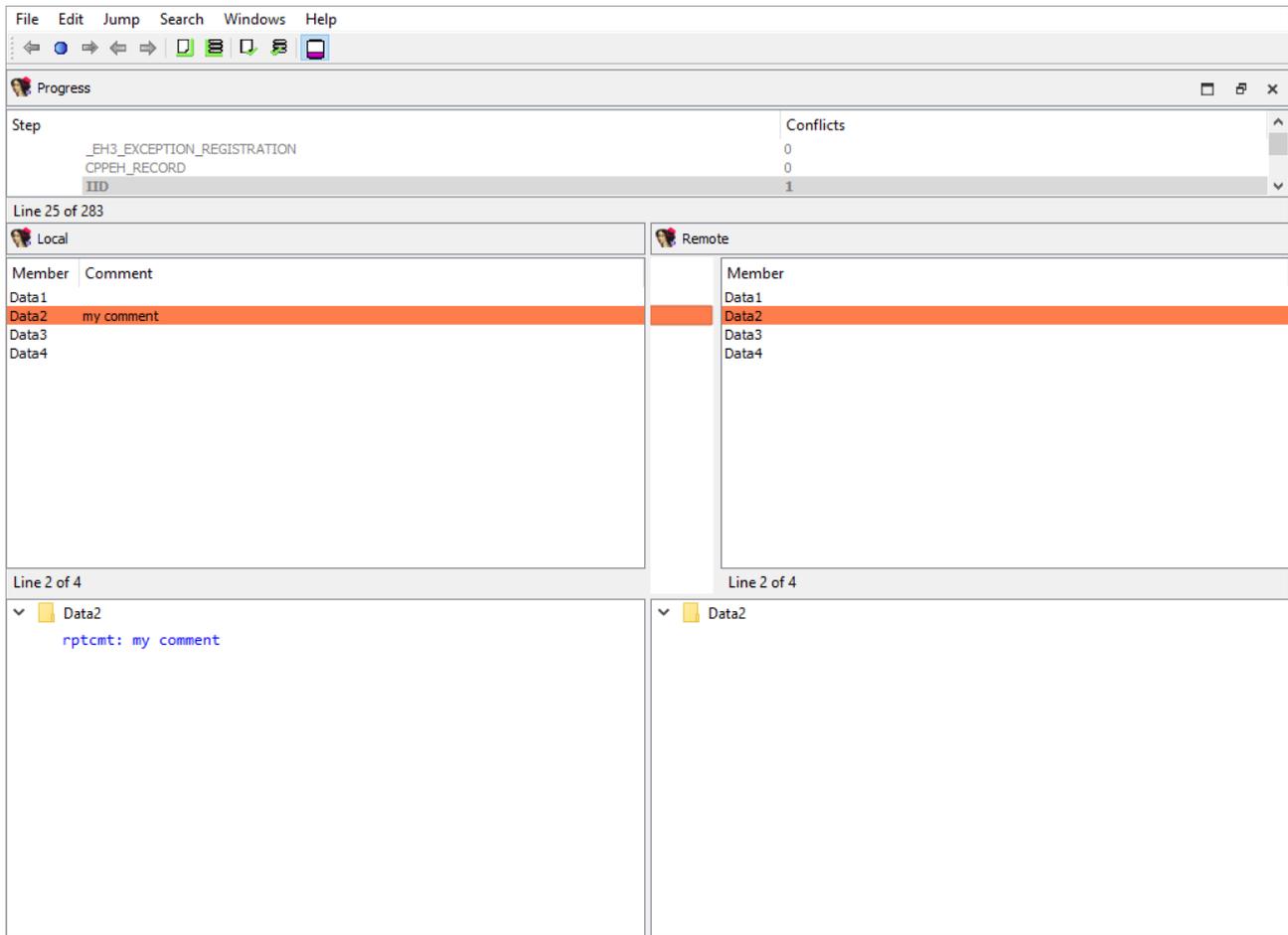
```
cmt: remote comment
```

We need a separate phase for these comments in order not to lose them during merging because by default ghost types are considered secondary to the corresponding non-ghost type. Normally during merge ghost types may be overwritten. However, local types cannot have comments at all. This is why ghost structure comments, if created, are valuable.

3.2.7. Types/Struct members comments/...

Similarly to comments attached to entire structures, each structure member may have a comment.

The same logic applies to ghost struct member comments:

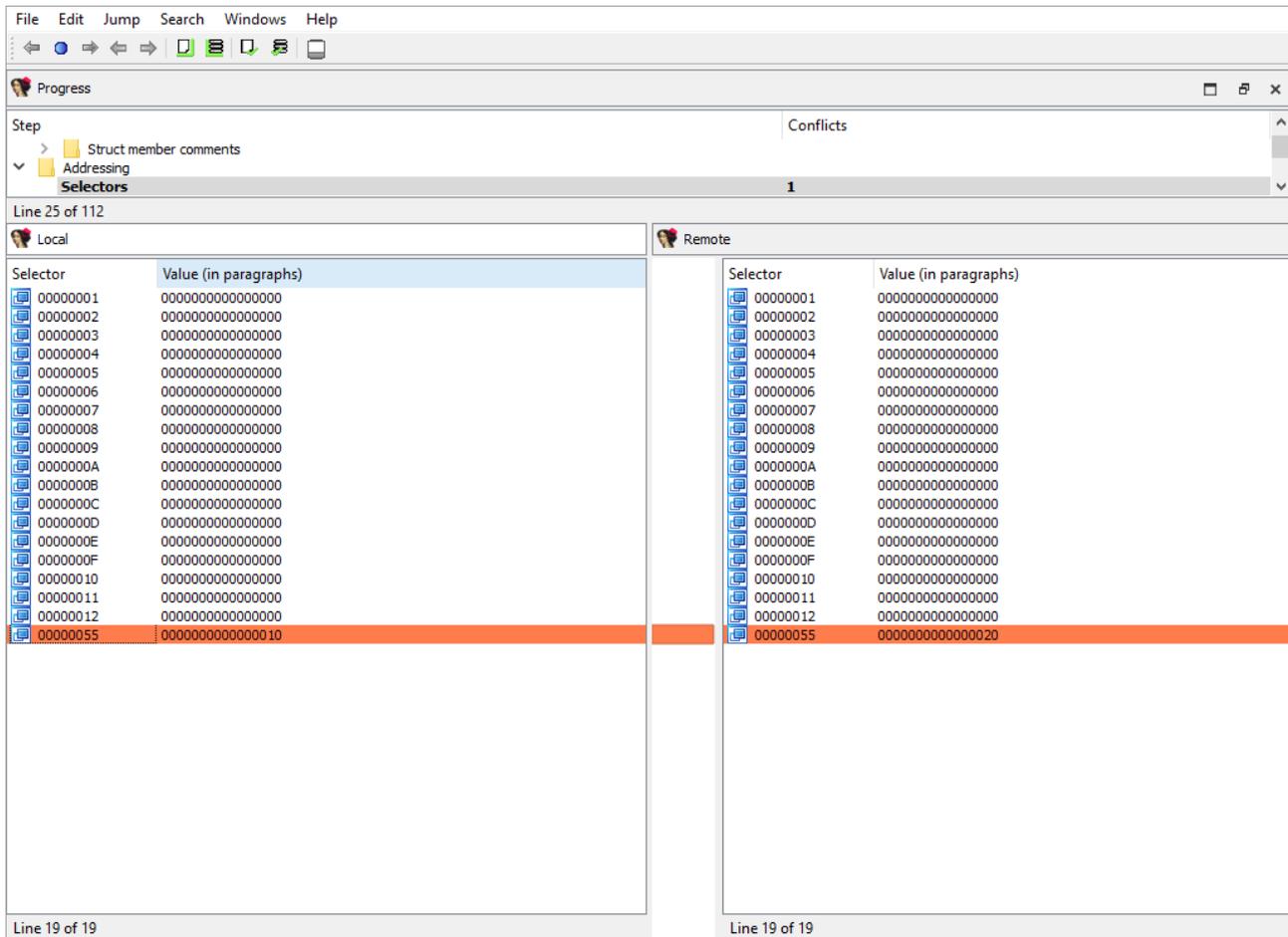


3.3. Addressing/Selectors

Merging of selectors.

This merge phase uses the standard widget "Selectors".

The "Detail" pane is absent.



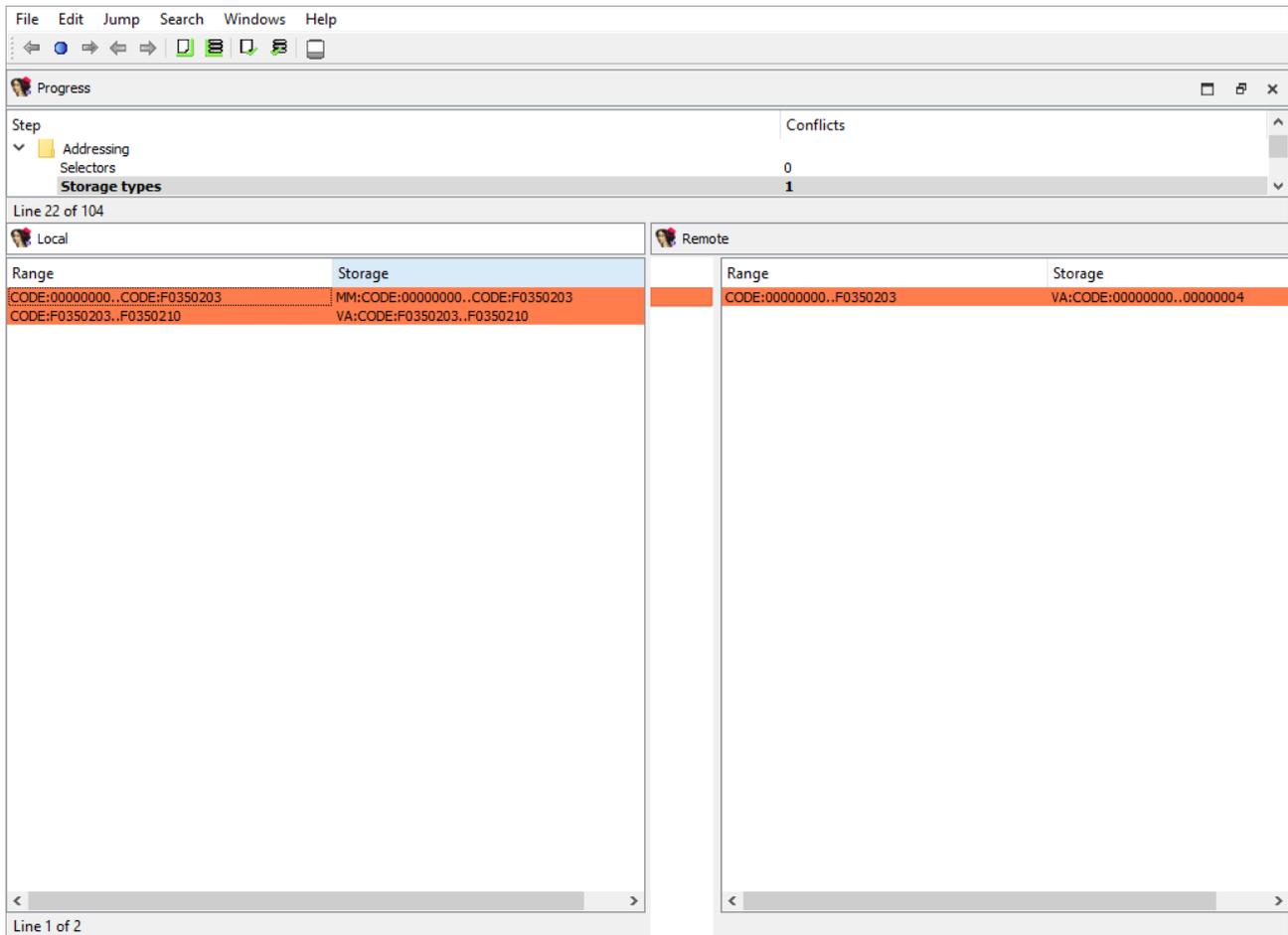
3.3.1. Addressing/Storage types

IDA Pro allocates so-called **flags** for each program address. These flags describe how to display the corresponding bytes in the disassembly listing: as instruction or data.

There are two different storage methods for **flags**: virtual array (VA) and sparse storage (MM). The virtual array method is the default one, it allocates 32 bits for each program address. However, for huge segments this method is not efficient and may lead to unnecessarily huge databases. Therefore for huge segments IDA Pro uses sparse storage.

This merge phase handles the defined program ranges and their storage types.

The "Detail" pane is absent.



3.3.2. Addressing/Segmentation

This merge phase handles the program segmentation.

When merging segments, IDA combines them into non-overlapping groups. Each group will have its own `diffpos`. For example, the following segmentations:

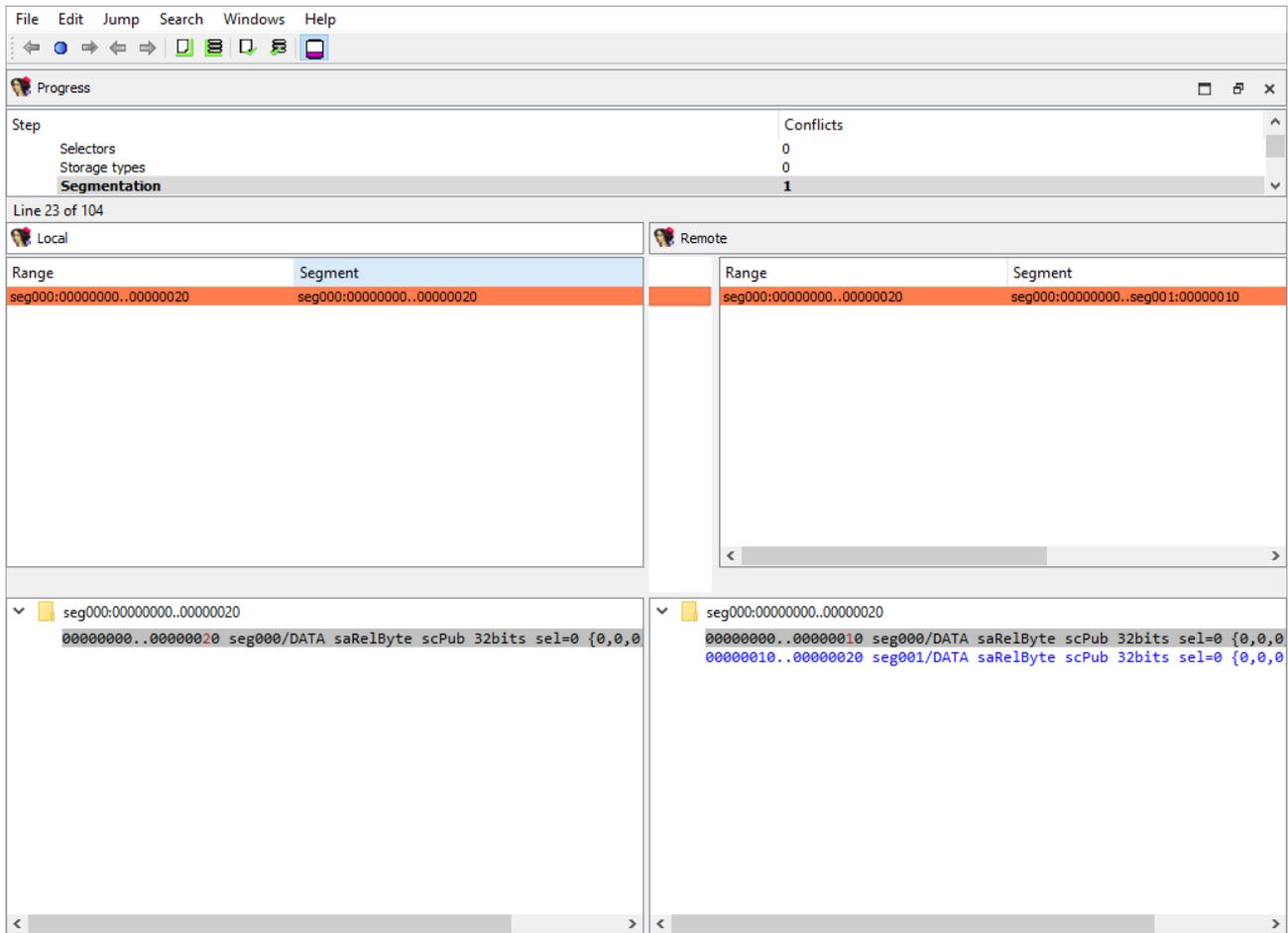
```

local_idb
  seg000:00000000
  ...
  seg000:00000020
  ...

remote_idb
  seg000:00000000
  ...
  seg001:00000010
  ...
  seg001:00000020

```

will result in a single `diffpos`:



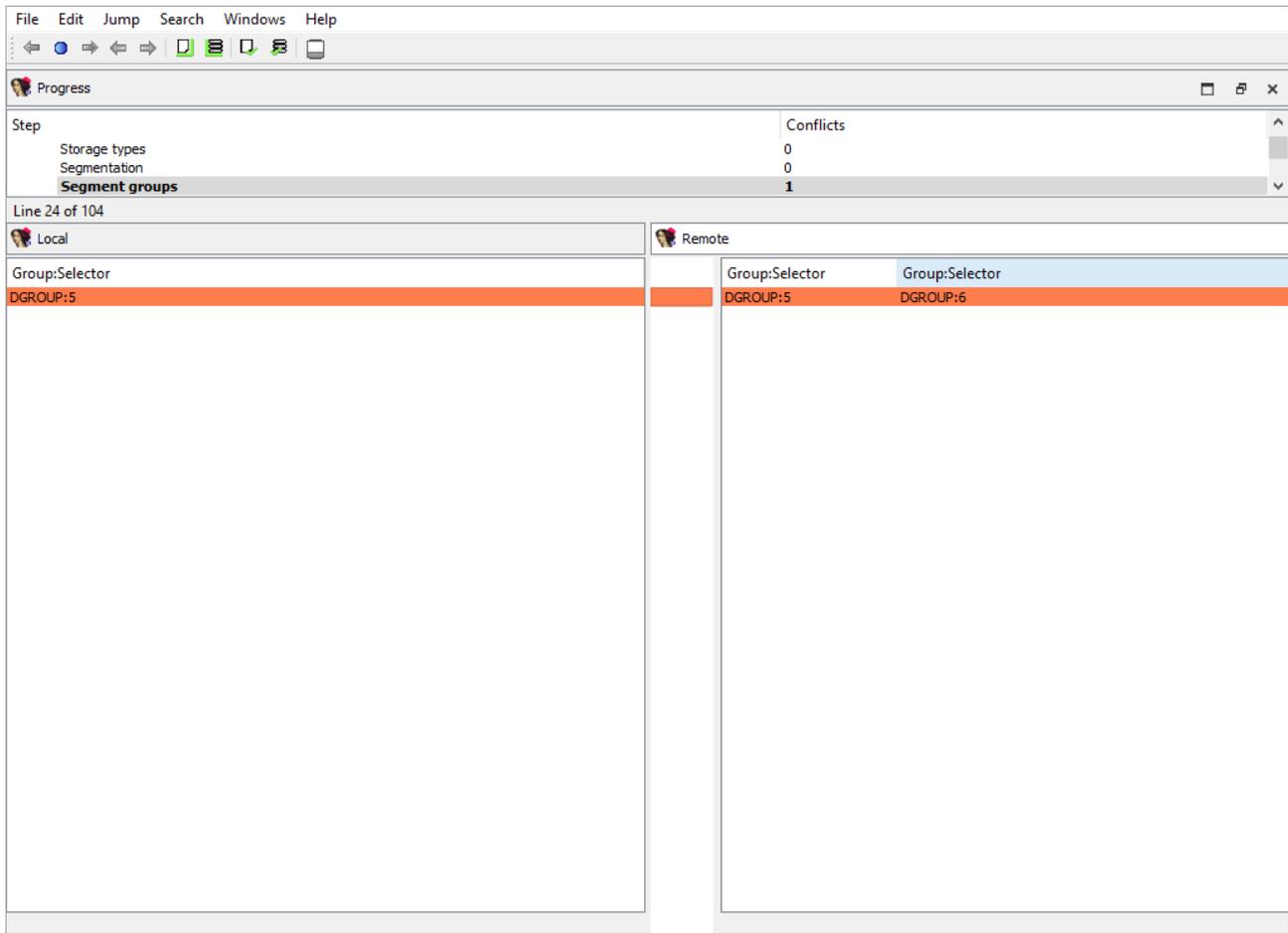
The "Detail" pane displays segments in the combined group with their attributes.

When merging segment, IDA tries to move the segment boundaries in a way that preserves the segment contents. If it fails to do so, the conflicting segments are deleted and new ones are created.

3.3.3. Addressing/Segment groups

Merging of segment groups. Segment groups are used only in OMF files. They correspond to the `group` keyword in assembler.

The "Detail" pane is absent.



3.3.4. Addressing/Segment register/...

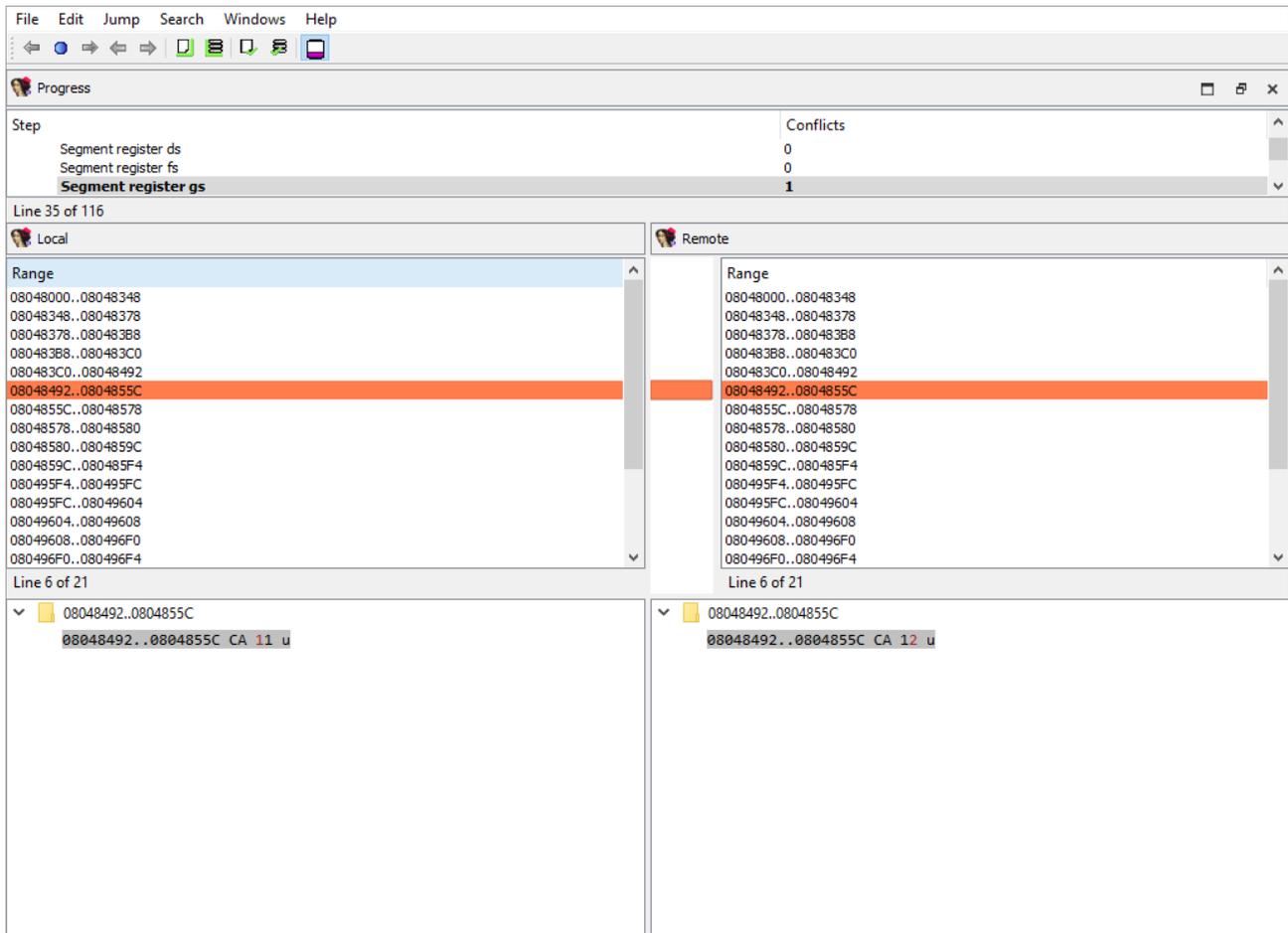
Some processor have so-called segment registers. IDA Pro knows about them and can remember their value (one value per address range).

For example, the x86 processor has *ds*, *ss*, and many other registers. IDA Pro can remember that, say, *ds* has the value of 1000 at the range 401000..402000.

This merge phase handles segment registers. For each register, a separate merge phase is created. It contains address ranges: inside each address range the value of the segment register stays the same.

To prepare *diffpos*, IDA Teams combines segment register ranges into non-overlapping ranges. *diffpos* is a range number.

The "Detail" pane displays segment register ranges in *diffpos* with the value and the suffix that denotes the range type (u-user defined, a-automatically inherited from the previous range)

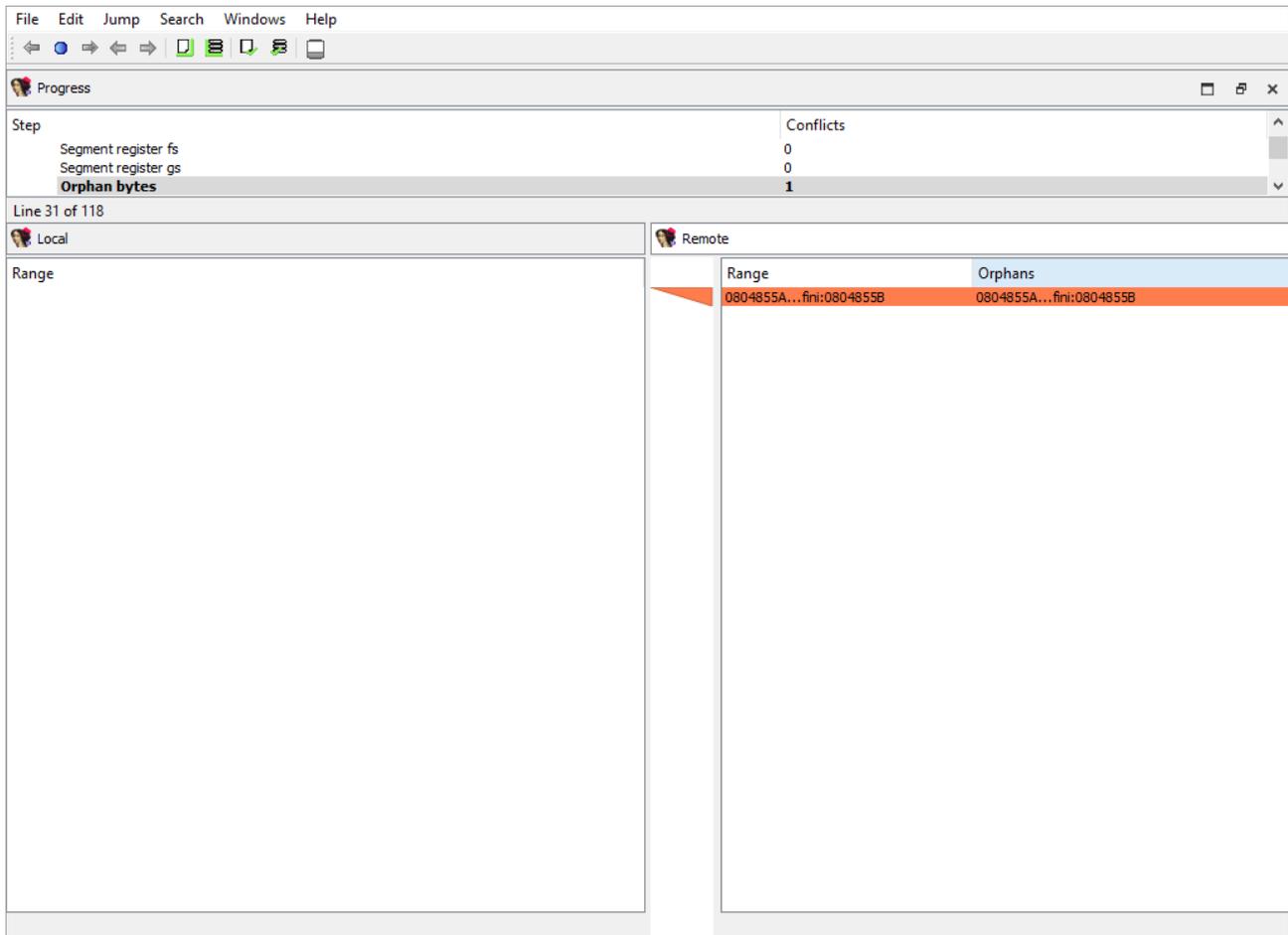


3.3.5. Addressing/Orphan bytes

The database may have bytes that do not belong to any segment.

To prepare `diffpos`, IDA Teams groups orphan bytes in the databases into nonintersecting ranges. `diffpos` is a range number.

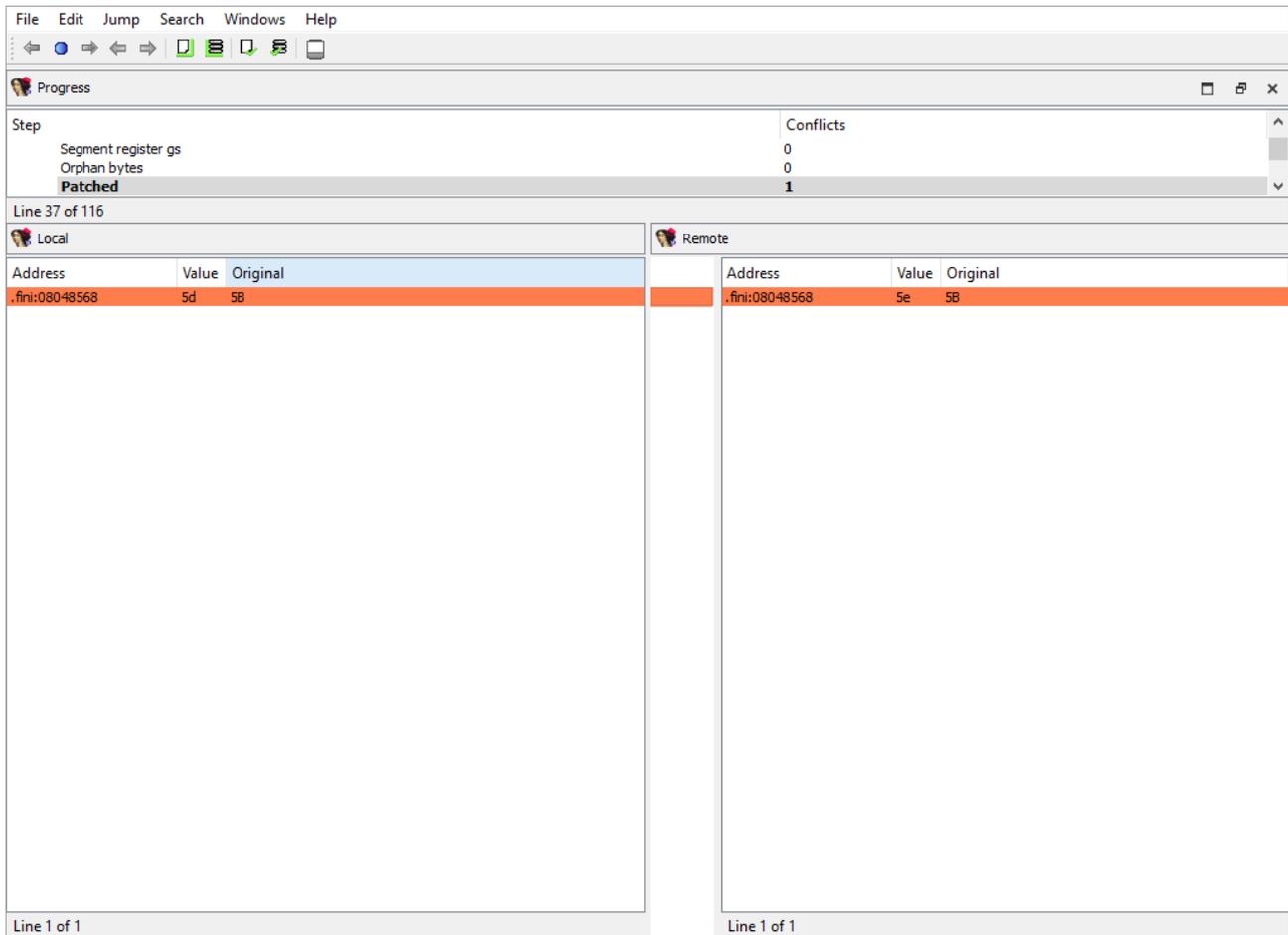
The "Detail" pane is absent.



3.3.6. Addressing/Patched

Merging of the patched bytes.

The "Detail" pane is absent.



3.3.7. Addressing/Byte values

Byte values in segments may differ even for non-patched addresses, for example if a snapshot of the process memory was taken during a debugger session.

IDA Teams combines the sequential bytes in one `diffpos`.

This merge phase uses the standard "IDA-View" widget.

The "Detail" pane displays the conflicting byte values.

File Edit Jump Search Windows Help

Progress

Step	Conflicts
Orphan bytes	0
Patched	0
Byte values	1

Line 30 of 100

Local	Remote
.text:081DD309 ??	.text:502D9312 2D 50 STR RS, [R5]
.text:081DD30A ??	.text:502D9314 45 90 STR R0, [SP]
.text:081DD30B ??	.text:502D9316 2D 50 STR RS, [R5]
.text:081DD30C ??	.text:502D9318 0D 8E LDRH RS, [R1]
.text:081DD30D ??	.text:502D931A 2D 50 STR RS, [R5]
.text:081DD30E ??	.text:502D931A
.text:081DD30F ??	.text:502D931C ?? ?? ?? ??... ;
.text:081DD310	.text:502D931C ?? ?? ?? ??... ; \$ 0xB8
.text:081DD310 CF FA ED FE CODE32	.text:502D93D4 ?? ?? ?? ??... dword_502D93D4 \$ 0xFFFFFFFFFA5F
.text:081DD310 DCD 0xFEEDEFAC	.text:502D93D4 ?? ?? ?? ??... ; .text ends
.text:081DD314 0C 00 00 01 DCD 0x100000C	.text:502D93D4 ?? ?? ?? ??... ;
.text:081DD318 00 00 00 00 DCD 0	.text:502D93D4 ?? ?? ?? ??... ;
.text:081DD31C 0B 00 00 00 DCD 0xB	.text:502D93D4 ?? ?? ?? ??... ;
.text:081DD320 02 00 00 00 DCD 2	.text:502D93D4 ?? ?? ?? ??... ;
.text:081DD324 B0 00 00 00 DCD 0xB0	.text:502D93D4 ?? ?? ?? ??... ;
.text:081DD328 01 00 00 00 DCD 1	.text:502D93D4 ?? ?? ?? ??... ;

00000000 FFFFFFFF081DD310: .text:081DD310

UNKNOWN 00000000502D93D4: .text:dword_502D93D4

FFFFFFF0081DD310..FFFFFFF0081DF670

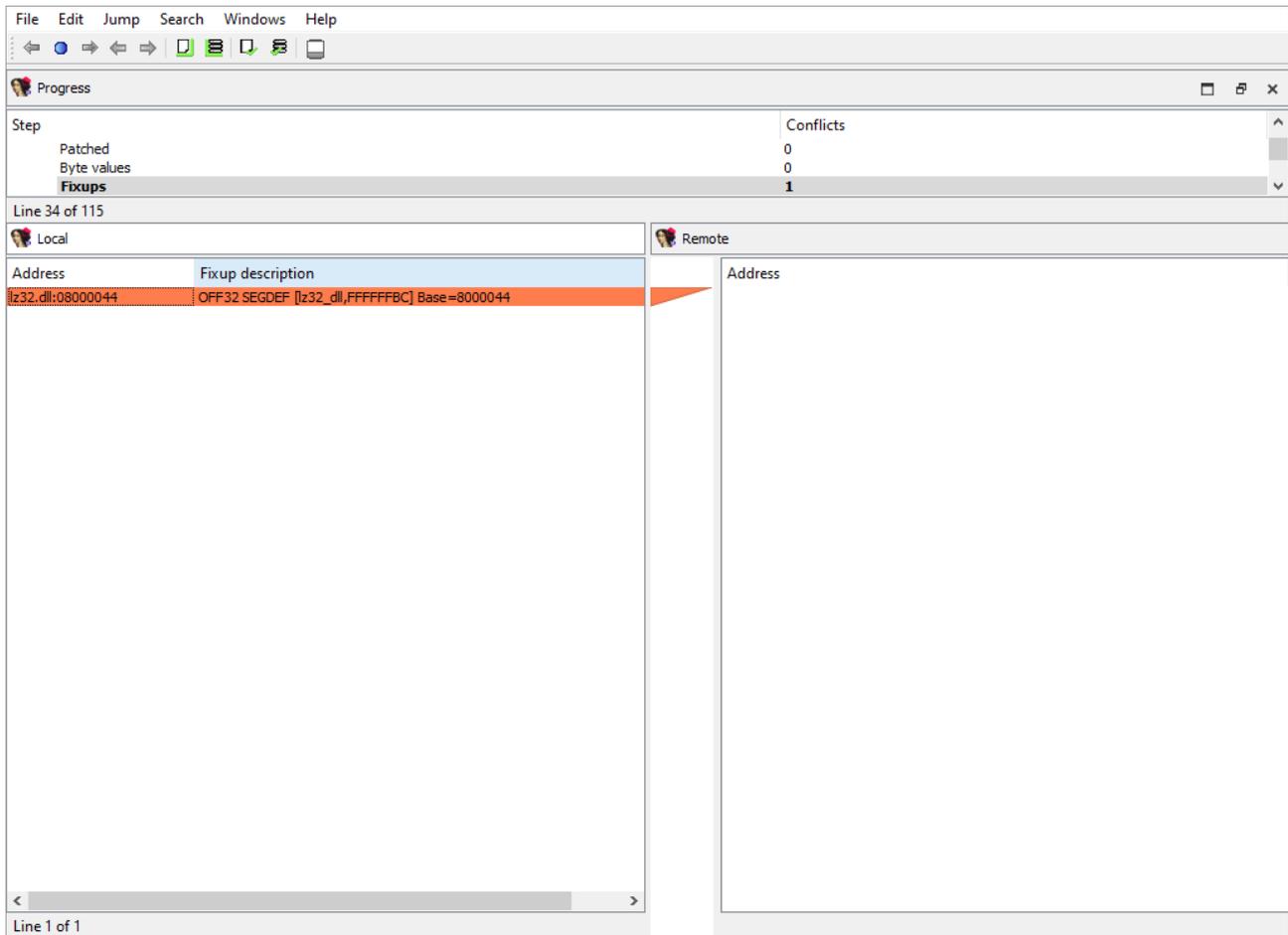
FFFFFFF0081DD310:CF
 FFFFFFFF0081DD311:FA
 FFFFFFFF0081DD312:ED
 FFFFFFFF0081DD313:FE
 FFFFFFFF0081DD314:0C
 FFFFFFFF0081DD315:00
 FFFFFFFF0081DD316:00
 FFFFFFFF0081DD317:01
 FFFFFFFF0081DD318:00
 FFFFFFFF0081DD319:00
 FFFFFFFF0081DD31A:00
 FFFFFFFF0081DD31B:00
 FFFFFFFF0081DD31C:0B
 FFFFFFFF0081DD31D:00
 FFFFFFFF0081DD31E:00
 FFFFFFFF0081DD31F:00
 FFFFFFFF0081DD320:02
 FFFFFFFF0081DD321:00

FFFFFFF0081DD310:??
 FFFFFFFF0081DD311:??
 FFFFFFFF0081DD312:??
 FFFFFFFF0081DD313:??
 FFFFFFFF0081DD314:??
 FFFFFFFF0081DD315:??
 FFFFFFFF0081DD316:??
 FFFFFFFF0081DD317:??
 FFFFFFFF0081DD318:??
 FFFFFFFF0081DD319:??
 FFFFFFFF0081DD31A:??
 FFFFFFFF0081DD31B:??
 FFFFFFFF0081DD31C:??
 FFFFFFFF0081DD31D:??
 FFFFFFFF0081DD31E:??
 FFFFFFFF0081DD31F:??
 FFFFFFFF0081DD320:??
 FFFFFFFF0081DD321:??

3.3.8. Addressing/Fixups

Merging of fixup records.

The "Detail" pane is absent.



3.3.9. Addressing/Manual memory mapping

Merging of memory mappings.

The "Detail" pane is absent.

The screenshot shows a debugger interface with a menu bar (File, Edit, Jump, Search, Windows, Help) and a toolbar. Below is a 'Progress' window with a table:

Step	Conflicts
Byte values	0
Fixups	0
Manual memory mapping	1

Below the progress window, there are two panels: 'Local' and 'Remote', each displaying a table of memory ranges and mappings.

Local Panel:

Range	Mapping
00000300..seg001:00000400	00000300..seg001:00000400 -> 00000200
seg001:00000400..00000500	seg001:00000400..00000500 -> 00000300

Remote Panel:

Range	Mapping
00000300..seg001:00000400	00000300..seg001:00000400 -> 00000200
seg001:00000400..00000500	seg001:00000400..00000500 -> 00000200
FFFFFFE0..FFFFFFF0	FFFFFFE0..FFFFFFF0 -> FFFFFFF0

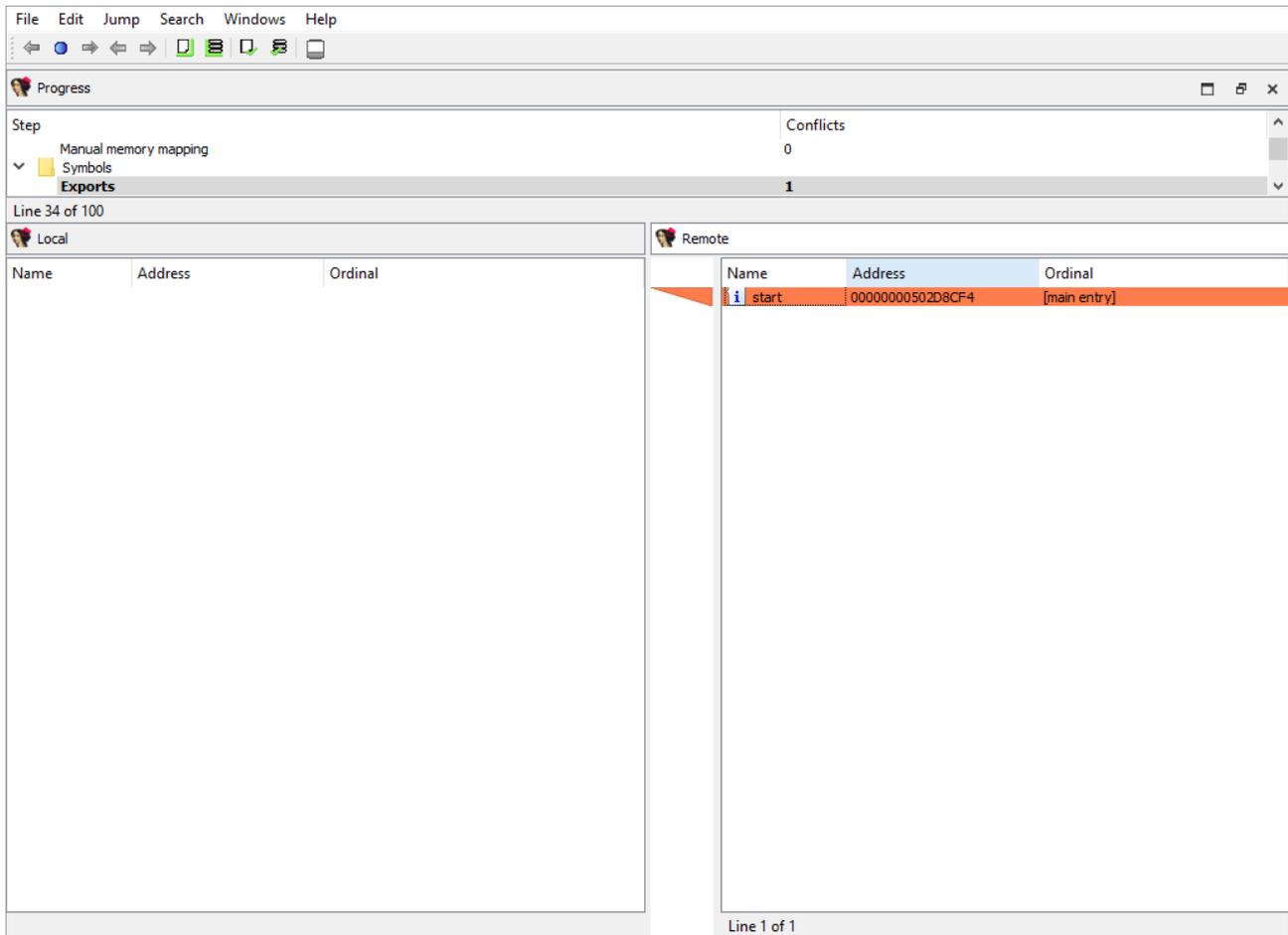
The status bars at the bottom indicate 'Line 2 of 2' for the Local panel and 'Line 2 of 3' for the Remote panel.

3.3.10. Symbols/Exports

Merging of exported symbols.

Merge phase uses the standard "Exports" widget.

The "Detail" pane is absent.



3.3.11. Symbols/Imports

Merging of imported symbols.

Merge phase uses the standard "Imports" widget.

The "Detail" pane is absent.

File Edit Jump Search Windows Help

Progress

Step Conflicts

Symbols Exports Imports 0 2

Line 39 of 112

Address	Ordinal	Name
000000000041574	291	
000000000041578		mbrtowc@@GLIBC_2.17
000000000041580		memcpy_local@@GLIBC_2.17
000000000041584	273	
000000000041588		memmove@@GLIBC_2.17
000000000041590		cap_to_text
000000000041598		_exit@@GLIBC_2.17
0000000000415A0		getcwd@@GLIBC_2.17
0000000000415A8		fwrite_unlocked@@GLIBC_2.17
0000000000415B0		strtol@@GLIBC_2.17
0000000000415B8		strnlen@@GLIBC_2.17
0000000000415C0		__sprintf_chk@@GLIBC_2.17
0000000000415C8		mbstowcs@@GLIBC_2.17
0000000000415D0		exit@@GLIBC_2.17
0000000000415D8		_setjmp@@GLIBC_2.17
0000000000415E0		raise@@GLIBC_2.17
0000000000415E8		error@@GLIBC_2.17
0000000000415F0		program_invocation_name@@GLIBC_2.17
0000000000415F8		sigprocmask@@GLIBC_2.17
000000000041600		localtime_r@@GLIBC_2.17
000000000041608		setenv@@GLIBC_2.17
000000000041610		readlink@@GLIBC_2.17
000000000041618		getgrnam@@GLIBC_2.17
000000000041620		__cxa_finalize@@GLIBC_2.17
000000000041628		opendir@@GLIBC_2.17
000000000041630		strftime@@GLIBC_2.17
000000000041638		__cxa_atexit@@GLIBC_2.17
000000000041640		iswcntrl@@GLIBC_2.17
000000000041648		clock_gettime@@GLIBC_2.17
000000000041650		stderr@@GLIBC_2.17
000000000041658		lseek@@GLIBC_2.17
000000000041660		__fpending@@GLIBC_2.17

Line 1 of 127

Address	Ordinal	Name
000000000041574	801	
000000000041578		mbrtowc@@GLIBC_2.17
000000000041580		memcpy@@GLIBC_2.17
000000000041584	546	
000000000041588		memmove@@GLIBC_2.17
000000000041590		cap_to_text
000000000041598		_exit@@GLIBC_2.17
0000000000415A0		getcwd@@GLIBC_2.17
0000000000415A8		fwrite_unlocked@@GLIBC_2.17
0000000000415B0		strtol@@GLIBC_2.17
0000000000415B8		strnlen@@GLIBC_2.17
0000000000415C0		__sprintf_chk@@GLIBC_2.17
0000000000415C8		mbstowcs@@GLIBC_2.17
0000000000415D0		exit@@GLIBC_2.17
0000000000415D8		_setjmp@@GLIBC_2.17
0000000000415E0		raise@@GLIBC_2.17
0000000000415E8		error@@GLIBC_2.17
0000000000415F0		program_invocation_name@@GLIBC_2.17
0000000000415F8		sigprocmask@@GLIBC_2.17
000000000041600		localtime_r@@GLIBC_2.17
000000000041608		setenv@@GLIBC_2.17
000000000041610		readlink@@GLIBC_2.17
000000000041618		getgrnam@@GLIBC_2.17
000000000041620		__cxa_finalize@@GLIBC_2.17
000000000041628		opendir@@GLIBC_2.17
000000000041630		strftime@@GLIBC_2.17
000000000041638		__cxa_atexit@@GLIBC_2.17
000000000041640		iswcntrl@@GLIBC_2.17
000000000041648		clock_gettime@@GLIBC_2.17
000000000041650		stderr@@GLIBC_2.17
000000000041658		lseek@@GLIBC_2.17
000000000041660		__fpending@@GLIBC_2.17

Line 1 of 127

3.3.12. Disassembly/Items

When merging, IDA Teams compares disassembly items (instructions and data). IDA Teams compares disassembly items by length, flags, opinfo, name, comment, and netnode information (NALT_* and NSUP_* flags).

This merge step uses the standard "IDA-View" widget so that items can be viewed in their context. For example:

The screenshot displays the IDA Pro interface during a merge phase. The main window is split into four panes:

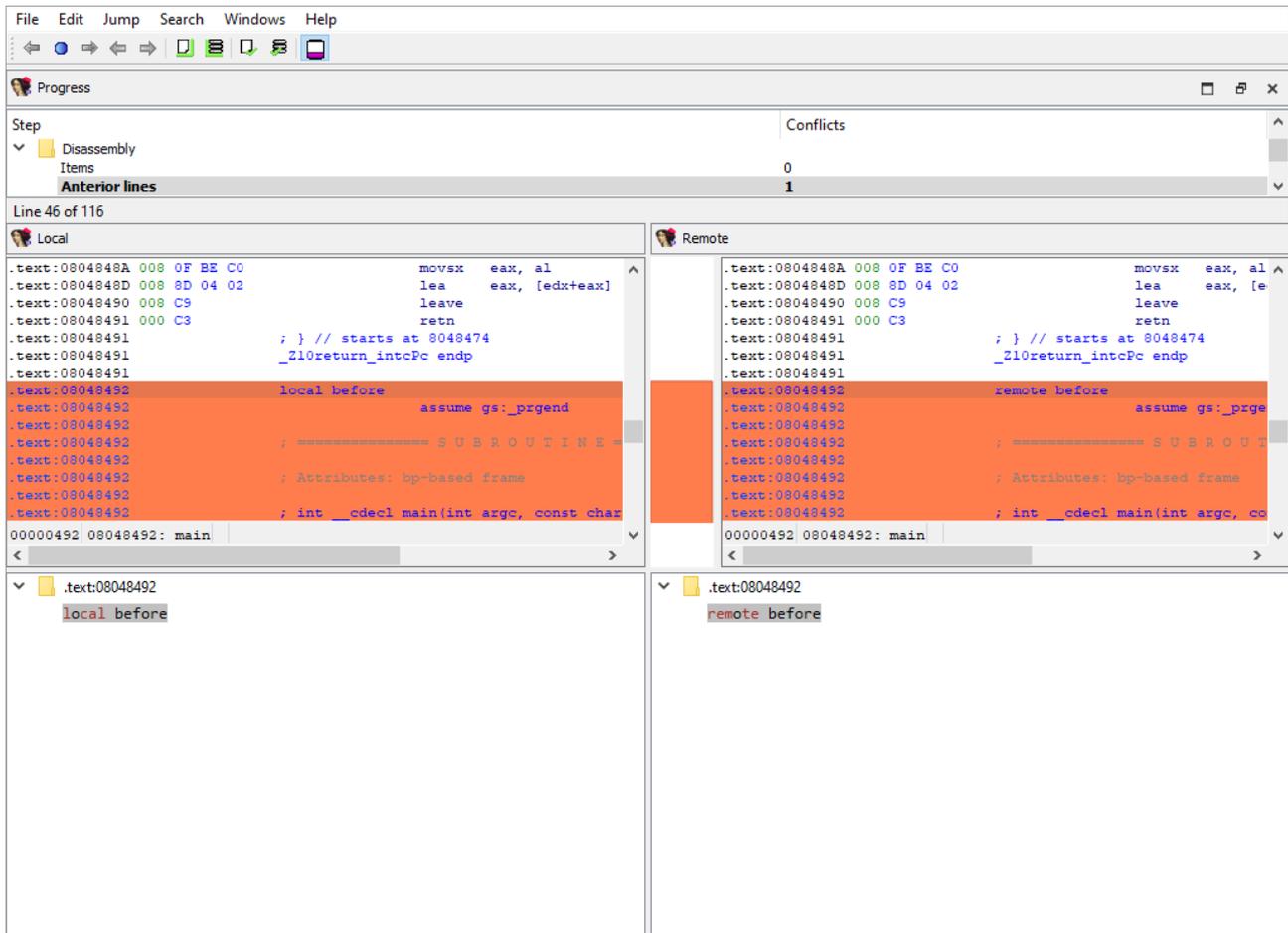
- Top Left (Local):** Shows assembly code for the local view. A label at address 0804849C is marked as `public label`.
- Top Right (Remote):** Shows assembly code for the remote view. The corresponding label at address 0804849C is marked as `local=[label]`.
- Bottom Left (Local Detail):** Shows the comment content for the selected label: `code:3`, `has_immd 1:stkvar username flow`, `1:stkvar`, and `public [label]`.
- Bottom Right (Remote Detail):** Shows the comment content for the selected label: `code:3`, `has_immd 1:stkvar username flow`, `1:stkvar`, and `local=[label]`.

3.3.13. Comments/Anterior lines and Comments/Posterior lines

Merging of extra comments.

This merge phase uses the standard "IDA-View" widget.

The "Detail" pane displays comment content.



3.3.14. Disassembly/EA additional flags

Merging of additional flags `aflags_t`.

Each disassembly item may have additional flags that further describe it.

This merge phase uses the standard "IDA-View" widget.

The "Detail" pane displays additional flags.

The screenshot displays the IDA Pro interface with a disassembly window split into 'Local' and 'Remote' panes. The assembly code is identical in both, showing instructions from address 0804849C to 080484B6. The instruction at 080484AF is highlighted in orange in both panes: `mov [ebp+foo], eax`. Below the assembly panes, the 'Detail' pane for the highlighted instruction shows the type `TI1` and the description `tif1: int`.

3.3.15. Disassembly/Hidden ranges

To prepare `diffpos`, IDA Teams groups hidden ranges into nonintersecting ranges. `diffpos` is a range number.

The "Detail" pane displays the hidden range description.

The screenshot displays the 'Hidden ranges' dialog in IDA Pro, comparing 'Local' and 'Remote' hidden ranges. The 'Progress' bar at the top shows 1 conflict. The 'Local' pane shows a range from .text:0000000000005A18 to .text:0000000000005A7C with a definition: 5A18..5A54 - 'dev_ino_hash_compare' and 5A54..5A7C + 'Local hiderange 2'. The 'Remote' pane shows the same range with a definition: 5A18..5A58 - 'dev_ino_hash+_compare' color FF0002 and 5A58..5A70 + 'sigHandler'.

3.3.16. Disassembly/Source file ranges

To prepare `diffpos`, IDA Teams groups source file ranges into nonintersecting ranges. `diffpos` is a range number.

The "Detail" pane displays source file definition.

The screenshot displays a software interface with a menu bar (File, Edit, Jump, Search, Windows, Help) and a toolbar. Below the menu is a "Progress" window showing a table of steps and conflicts:

Step	Conflicts
File regions	0
Hidden ranges	0
Source file ranges	1

Below the progress window, there are two side-by-side panes labeled "Local" and "Remote". Each pane has a table with "Range" and "File range" columns. The "Local" pane shows a range from `.text:08048474...text:080484C0` to `.text:08048474...text:08048492`. The "Remote" pane shows a range from `.text:08048474...text:080484C0` to `.text:08048474...text:080484C0`. Below these tables, there are two code editors. The "Local" editor shows a list of file ranges with corresponding file names: `8048474..8048492 local_file.c` and `8048492..80484C0 local2_file.c`. The "Remote" editor shows a list of file ranges with corresponding file names: `8048474..80484C0 remote_file.c`.

3.3.17. Functions/Registry

Function definitions (`func_t`) are merged using the standard "Functions" widget, while the "Detail" pane displays function attributes:

The screenshot displays the IDA Pro interface with two function lists and their corresponding detail panes.

Local View:

Function name	Segment	Start	Length	Locals
f test1(C *)	.text	00000000	0000021B	0000026C
f B1::a1(void)	.text	000006A0	00000015	0000001C
f B6::~B6()	.text\$_ZN2B6...	00000DB4	00000055	0000001C
f B6::~B6()	.text\$_ZN2B6...	00000E0C	00000026	0000001C
f A6::A6(void)	.text\$_ZN2A6...	00000E34	00000010	00000004
f B7::~B7(void)	.text\$_ZN2B7...	00000E44	00000073	0000001C
f B7::~B7()	.text\$_ZN2B7...	00000E88	00000063	0000001C
f B7::~B7()	.text\$_ZN2B7...	00000F1C	00000063	0000001C
f B7::~B7()	.text\$_ZN2B7...	00000F80	00000026	0000001C
f A7::A7(void)	.text\$_ZN2A7...	00000FA8	00000010	00000004
f C::C(void)	.text\$_ZN1C...	00000FB8	00000081	0000001C
f D1::D1(void)	.text\$_ZN2D1...	0000104C	0000001E	0000001C
f D1::~D1()	.text\$_ZN2D1...	0000106C	0000001E	0000001C
f D1::~D1()	.text\$_ZN2D1...	0000108C	0000001E	0000001C
f D1::~D1()	.text\$_ZN2D1...	000010AC	00000026	0000001C

Remote View:

Function name	Segment	Start	Length	Local
f test1(C *)	.text	00000000	00000138	00000
f test2(void)	.text	00000138	000000E0	00000
f test3(D7 *)	.text	00000218	000000CF	00000
f test5(E7 *)	.text	000002E7	000000BF	00000
f test7(F1 *)	.text	000003A6	00000018	00000
f test8(F2 *)	.text	000003BE	00000018	00000
f test9(F3 *)	.text	000003D6	0000002B	00000
f test10(F4 *)	.text	00000401	0000003D	00000
f test11(F6 *)	.text	0000043E	0000003D	00000
f _main	.text	0000047B	0000023D	00000
f A1::f1(void)	.text\$_ZN2A1...	000006C0	0000000A	00000
f A2::f2(void)	.text\$_ZN2A2...	000006CC	0000000A	00000
f A3::f3(void)	.text\$_ZN2A3...	000006D8	0000000A	00000
f A4::f4(void)	.text\$_ZN2A4...	000006E4	0000000A	00000
f A5::f5(void)	.text\$_ZN2A5...	000006F0	0000000A	00000

Local Detail Pane (Line 1 of 85):

```

_Z5test1P1C
start_ea: 0
end_ea : 21B
Attributes: bp-based frame prolog-analysis-ok purged-analysis-ok
lvars-size: 268
saved-regs: 4
npurged : 0

```

Remote Detail Pane (Line 1 of 154):

```

_Z5test1P1C
start_ea: 0
end_ea : 138
Attributes: bp-based frame sp-ready prolog-analysis-ok purged-anal
lvars-size: 18
saved-regs: 4
npurged : 0

```

3.3.18. Functions/IM flags

Merging of instruction kinds.

To simplify decompilation, IDA has the notion of the instruction kind:

- PROLOG instruction
- EPILOG instruction
- SWITCH instruction

This merge phase uses the standard "IDA-View" widget.

The "Detail" pane displays instruction kind.

The screenshot displays a debugger's merge phase for function frames. The interface is divided into four main sections:

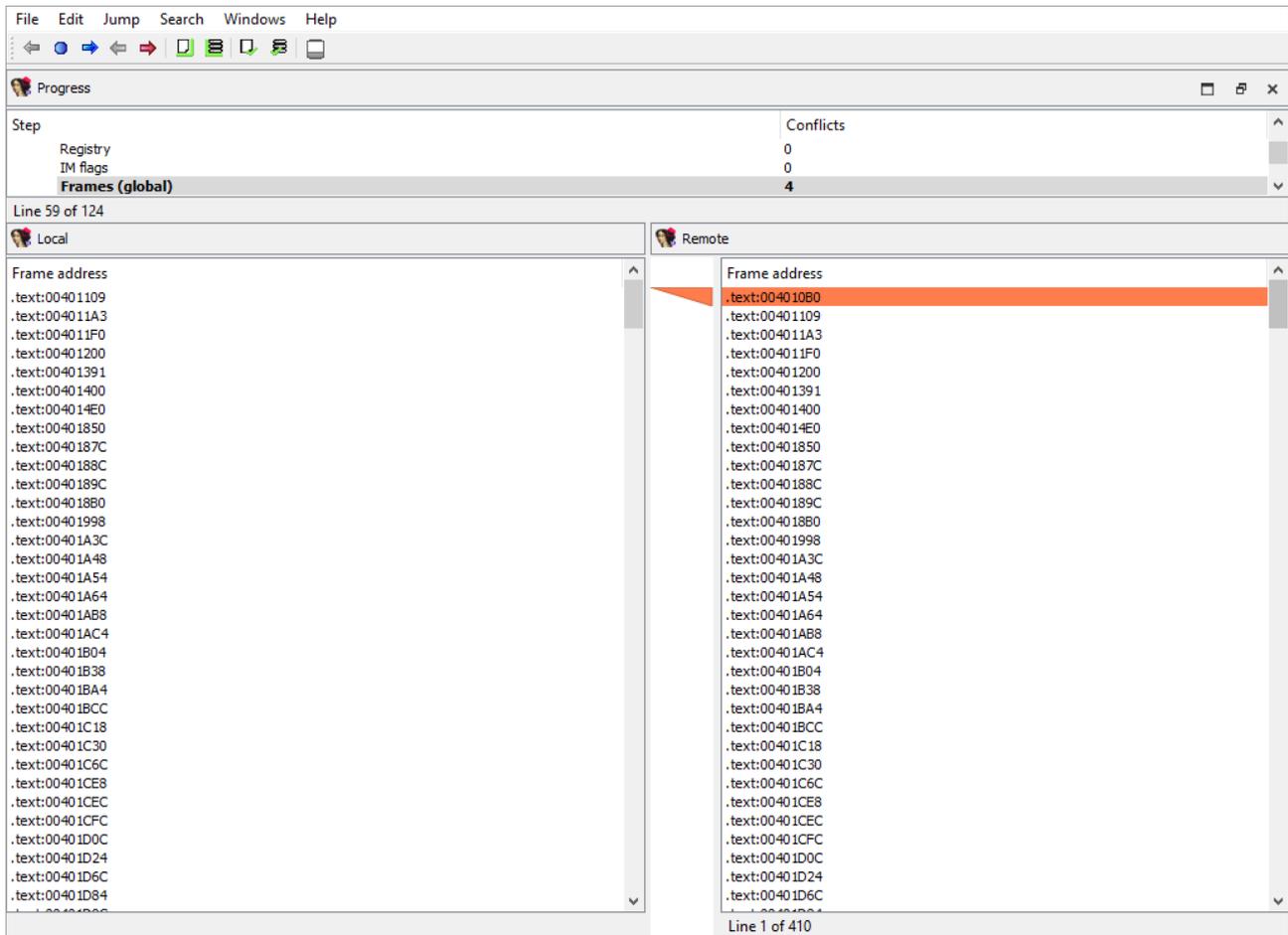
- Progress:** Shows the current step as 'IM flags' with 2 conflicts.
- Local:** Contains assembly code for the local frame. The code includes variable declarations for `in_char`, `arg_0`, and `in_char_ptr`, followed by stack frame setup (pushing `ebp` and subtracting 4 from `esp`). The code then moves `eax` to `edx` and performs various `movsx` and `lea` operations. The function signature is `return_int(char, char *)+3`.
- Remote:** Contains assembly code for the remote frame, which is identical to the local frame.
- EPILOG:** Labeled as `.text:08048477`, this pane is currently empty.
- PROLOG:** Labeled as `.text:08048477`, this pane is currently empty.

3.3.19. Functions/Frames (global)

This merge phase deals with the entire function frames. Function frame may be added or deleted.

If members of the matched function frame differ, the conflict will be resolved later during the **Functions/Frame/...** merge phase. Each differing frame will be assigned its own merge step.

The "Detail" pane is absent.



3.3.20. Functions/Frame

Merging of function frame details.

A separate phase is created for each function. For example:

- **Functions/Frames/sub_401200 at 401200**
- **Functions/Frames/_main at 4014E0**

Every of these phases merges the conflicting members of the function frame.

The "Detail" pane displays the detailed information about the current function frame member.

The screenshot displays the IDA Pro interface during a merge phase. At the top, the menu bar includes File, Edit, Jump, Search, Windows, and Help. Below the menu is a toolbar with navigation icons. The main window is divided into several panes:

- Progress:** Shows the current step and conflicts. The step is 'main at 4014E0' with 1 conflict.
- Local:** A table showing the local function's SP change points.

Range	Member
C..10	suffix
10..14	code
14..18	"s"
18..1C	"r"
1C..20	argc
20..24	argv
24..28	envp
- Remote:** A table showing the remote function's SP change points.

Range	Member
C..10	Source
10..14	var_4
14..18	"s"
18..1C	"r"
1C..20	argc
20..24	argv
24..28	envp
- Detail (Local):** Shows details for the selected range C..10.


```
"suffix" soff:C size:4
  flags:25500400 dword offset
  > 10..14
```
- Detail (Remote):** Shows details for the selected range C..10.


```
"Source" soff:C size:4
  flags:25500400 dword offset
  TI
  tinfo:char *
  > 10..14
```

3.3.21. Functions/SP change points

Merging of function SP change points.

This merge phase uses the standard "IDA-View" widget.

The "Detail" pane displays the SP change point details.

The screenshot displays the IDA Pro interface with two side-by-side assembly windows: 'Local' and 'Remote'. Both windows show the same assembly code for the function 'sub_40C1A6'. The instruction at address 0040C1B4 is highlighted in orange in both views. In the 'Local' view, the instruction is `cmp [ebp+arg_0], 0` and the 'Detail' pane below it shows 'user-defined SP value: No'. In the 'Remote' view, the instruction is `cmp [ebp+arg_0], 0` and the 'Detail' pane shows 'user-defined SP value: Yes'. The 'Progress' window at the top indicates 'SP change points' with a count of 1. The assembly code includes instructions for setting up the stack frame, pushing arguments, and performing a comparison before jumping to `loc_40C623`.

3.3.22. Cross-references/Flow

Merging of regular execution flow from the previous instruction. IDA stores cross-references that correspond to regular execution flow in a special format, different from other cross-reference types.

This merge phase uses the standard "IDA-View" widget.

The "Detail" pane is absent.

The screenshot displays the IDA Pro interface with the 'Flow' pane selected. The 'Local' pane shows the assembly code for the local function, and the 'Remote' pane shows the assembly code for the remote function. The 'Detail' pane shows the code references to address (diffpos).

```

Local
.text:000005BF 1C0 00
.text:000005C4 1C0 8D 84 24 68      lea     eax, [esp+1BCh+
.text:000005C4 1C0 01 00 00
.text:000005CB 1C0 89 04 24      mov     [esp], eax
.text:000005CE 1C0 E8 5D 13 00 .rrrr   call   __2N2F3D1Ev
.text:000005CE 1C0 00
.text:000005D3 1C0 8D 84 24 9C      lea     eax, [esp+1BCh+
.text:000005D3 1C0 01 00 00
.text:000005DA 1C0 89 04 24      mov     [esp], eax
.text:000005DD 1C0 E8 52 16 00 .rrrr   call   __2N2F1D1Ev
.text:000005DD 1C0 00
.text:000005E2 1C0 C7 44 24 20      mov     [esp+1BCh+fcx.
.text:000005E2 1C0 FF FF FF FF      call   __25test2v
.text:000005EA 1C0 E8 49 FB FF .rrrr   call   __25tes
.text:000005EA 1C0 FF
.text:000005EF 1C0 89 44 24 18      mov     [esp+18h], eax
.text:000005F3 1C0 E9 A8 00 00      jmp     loc_6A0
.text:000005F3 1C0 00
.text:000005F8
.text:000005F8      loc_5F8:
.text:000005F8 1C0 8B 54 24 24      mov     edx, [esp+1BCh+
.text:000005FC 1C0 8B 44 24 20      mov     eax, [esp+1BCh+
.text:00000600 1C0 85 C0      test    eax, eax
.text:00000602 1C0 74 0C      jz      short loc_610
.text:00000604 1C0 83 E8 01      sub     eax, 1
.text:00000607 1C0 85 C0      test    eax, eax
.text:00000609 1C0 74 29      jz      short loc_634
.text:0000060B 1C0 83 E8 01      sub     eax, 1
.text:0000060E 1C0 0F 0B      ud2
.text:00000610
.text:00000610      loc_610:
.text:00000610 1C0 89 54 24 18      mov     [esp+18h], edx
.text:00000614 1C0 8D 44 24 50      lea     eax, [esp+1BCh+
.text:00000618 1C0 89 04 24      mov     [esp], eax
.text:0000061B 1C0 E8 4C 21 00 .rrrr   call   __2N2D7D1Ev
00002EFC 000005F8: _main:loc_5F8

Remote
.text:000005BF 1C0 00
.text:000005C4 1C0 8D 84 24 68      lea     eax, [e
.text:000005C4 1C0 01 00 00
.text:000005CB 1C0 89 04 24      mov     [esp],
.text:000005CE 1C0 E8 5D 13 00 .rrrr   call   __2N2F3
.text:000005CE 1C0 00
.text:000005D3 1C0 8D 84 24 9C      lea     eax, [e
.text:000005D3 1C0 01 00 00
.text:000005DA 1C0 89 04 24      mov     [esp],
.text:000005DD 1C0 E8 52 16 00 .rrrr   call   __2N2F1
.text:000005DD 1C0 00
.text:000005E2 1C0 C7 44 24 20      mov     [esp+1B
.text:000005E2 1C0 FF FF FF FF      call   __25tes
.text:000005EA 1C0 E8 49 FB FF .rrrr   call   __25tes
.text:000005EA 1C0 FF
.text:000005EF 1C0 89 44 24 18      mov     [esp+18
.text:000005F3 1C0 E9 A8 00 00      jmp     loc_6A0
.text:000005F3 1C0 00
.text:000005F8
.text:000005F8      loc_5F8:
.text:000005F8      ; cleanup() // owned by 4B7
.text:000005F8 1C0 8B 54 24 24      mov     edx, [e
.text:000005FC 1C0 8B 44 24 20      mov     eax, [e
.text:00000600 1C0 85 C0      test    eax, ea
.text:00000602 1C0 74 0C      jz      short 1
.text:00000604 1C0 83 E8 01      sub     eax, 1
.text:00000607 1C0 85 C0      test    eax, ea
.text:00000609 1C0 74 29      jz      short 1
.text:0000060B 1C0 83 E8 01      sub     eax, 1
.text:0000060E 1C0 0F 0B      ud2
.text:00000610
.text:00000610      loc_610:
.text:00000610 1C0 89 54 24 18      mov     [esp+18
.text:00000614 1C0 8D 44 24 50      lea     eax, [e
00002EFC 000005F8: _main:loc_5F8

```

3.3.23. Cross-references/Code

Merging of code cross-references.

This merge phase uses the standard "IDA-View" widget.

The "Detail" pane displays code references to address (diffpos).

The screenshot displays a debugger's cross-references window. The 'Step' window shows 0 conflicts and 4 cross-references. The 'Code' window is split into 'Local' and 'Remote' views, both showing assembly instructions. The instruction at address 000004C3 is highlighted in orange in both views. The 'Detail' panes at the bottom show data references for this instruction: the left pane shows a reference to `B6::~B6()+34/p` and the right pane shows a reference to `D7::D7(void)/p`.

3.3.24. Cross-references/Data

Merging of data cross-references.

This merge phase uses the standard "IDA-View" widget.

The "Detail" pane displays data references to address (`diffpos`).

The screenshot displays a debugger window with a menu bar (File, Edit, Jump, Search, Windows, Help) and a toolbar. Below the menu is a 'Progress' bar. The main area is divided into several panes:

- Step:** Shows 'Flow', 'Code', and 'Data' with 'Conflicts' counts (0, 0, 6).
- Line 65 of 120:** Indicates the current line of code.
- Local:** Contains assembly instructions. A red highlight is on the instruction:


```
.text$_ZN2B4D1Ev:00000B73 01C BA 38 22 00 .xxxx mov edx, [eax]
```
- Remote:** Contains the same assembly instructions as the Local pane. A red highlight is on the instruction:


```
.text$_ZN2B4D1Ev:00000B73 01C BA 38 22 00 .xxxx mov edx, [eax]
```
- Detail:** This pane is absent, showing only a tree view of memory locations:
 - Local: `.text$_ZN2B4D1Ev:00000B73` (expanded) showing `B5::B5(void)/o` and `.text$_ZN2B4D1Ev:00000B7E`.
 - Remote: `.text$_ZN2B4D1Ev:00000B73` (expanded) showing `.rdata$_ZTV2B4:off_2238/o` and `.text$_ZN2B4D1Ev:00000B7E`.

3.3.25. Marked positions/...

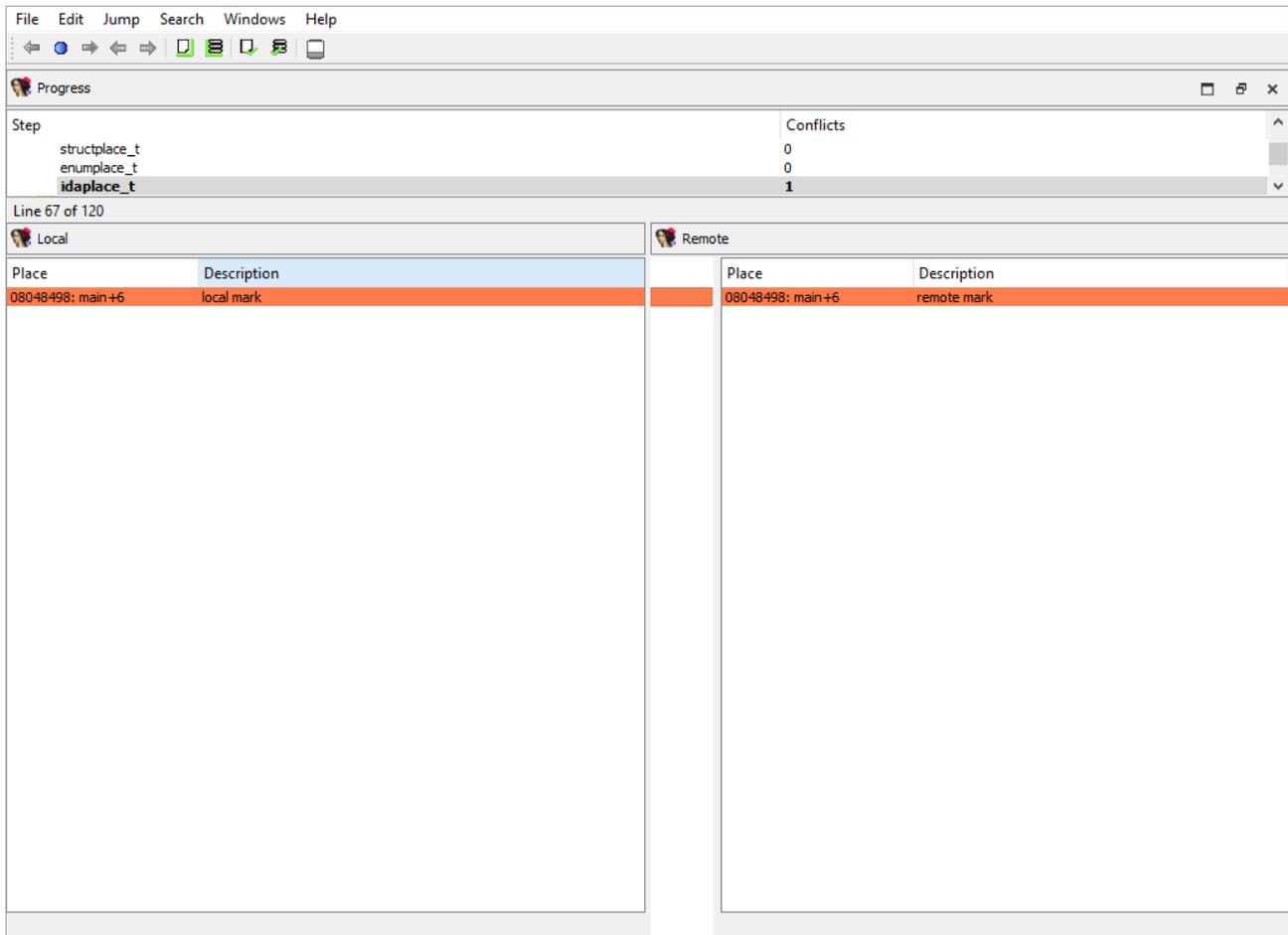
The following merge phases exist:

- **Marked positions/structplace_t**
- **Marked positions/enumplace_t**
- **Marked position/idaplace_t**

They deal with merging of bookmarks for:

- structures
- enums
- addresses

The "Detail" pane is absent.



3.3.26. Debug/Breakpoints/...

The following merge phases exist:

- **Breakpoints/Absolute bpts**
- **Breakpoints/Relative bpts**
- **Breakpoints/Symbolic bpts**
- **Breakpoints/Source level bpts**

They deal with merging of various debugger breakpoints.

The "Detail" pane is absent.

The screenshot displays a debugger's breakpoint management window. At the top, there is a menu bar with 'File', 'Edit', 'Jump', 'Search', 'Windows', and 'Help'. Below the menu is a toolbar with navigation icons. The main area is divided into several panes:

- Progress:** Shows 'Step' and 'Conflicts'.
- Breakpoints:** A tree view showing 'Debug' > 'Breakpoints' > 'Absolute bpts' with a count of '1'.
- Local:** A table listing local breakpoints.

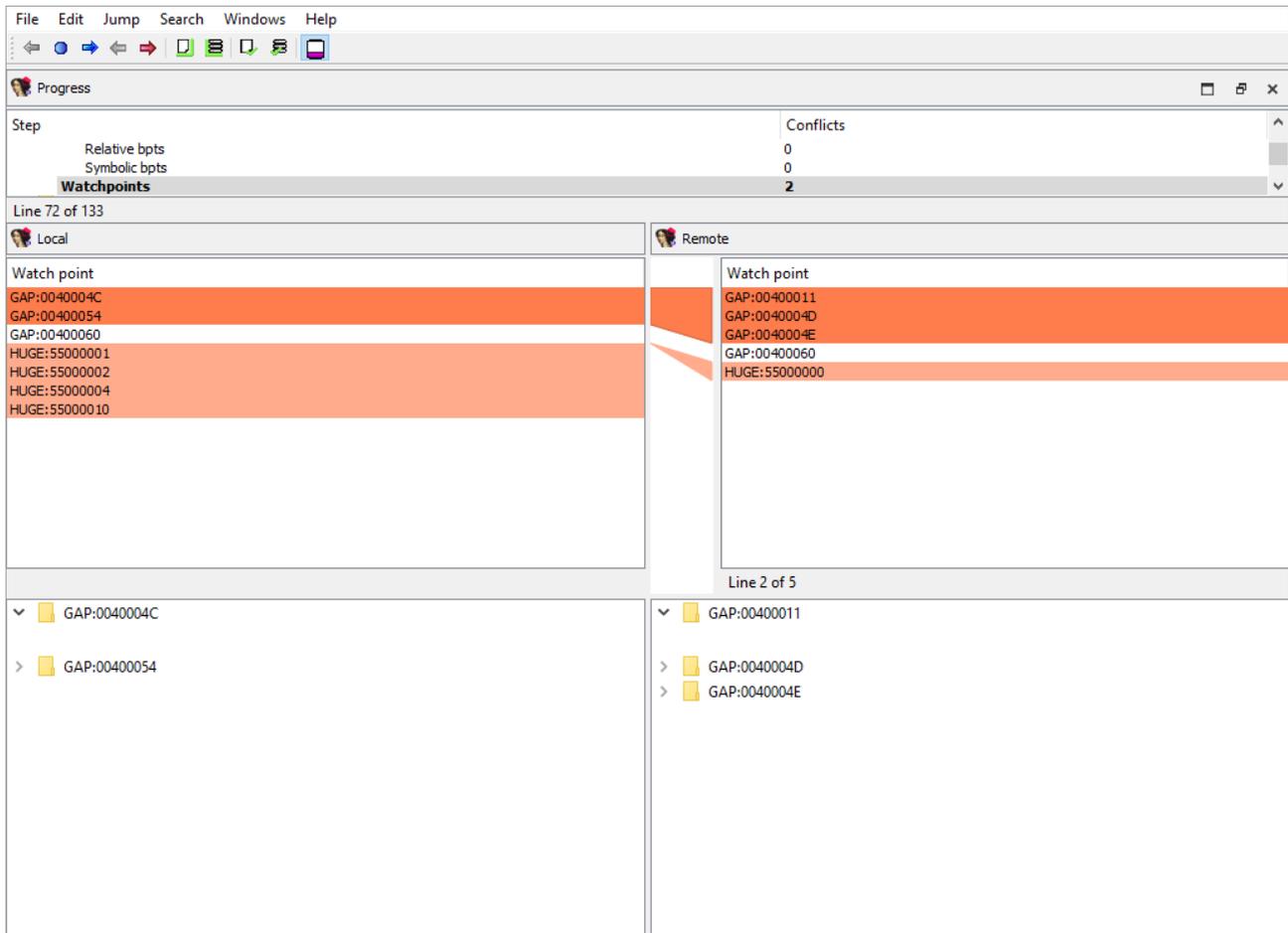
Range	Bpt	Bpt
0040000C..0040000D		
0040000F..00400010	0x40000F (mystart+3)	
00400016..0040002C	0x400016 (GAP:00400016)	0x400020 (GAP:remote_...)
- Remote:** A table listing remote breakpoints.

Range	Bpt
0040000C..0040000D	0x40000C (mystart)
0040000F..00400010	
00400016..0040002C	0x400018 (GAP:remote_name)
- Detail (Local):** Shows 'Line 2 of 3' with a tree view of breakpoints:
 - 0040000C..0040000D
 - > 0040000F..00400010
 - > 00400016..0040002C
- Detail (Remote):** Shows 'Line 2 of 3' with a tree view of breakpoints:
 - 0040000C..0040000D
 - 0x40000C (mystart)
 - Soft
 - Enabled
 - Break
 - EAX==10
 - > 0040000F..00400010
 - > 00400016..0040002C

3.3.27. Debug/Watchpoints

Merging of watch points.

The "Detail" pane is absent.



3.3.28. Dirtree/\$ dirtree/...

The following merge phases exist:

- **Dirtree/\$ dirtree/tinfos**
- **Dirtree/\$ dirtree/structs**
- **Dirtree/\$ dirtree/enums**
- **Dirtree/\$ dirtree/funcs**
- **Dirtree/\$ dirtree/names**
- **Dirtree/\$ dirtree/imports**
- **Dirtree/\$ dirtree/bookmarks_idaplace_t**
- **Dirtree/\$ dirtree/bookmarks_structplace_t**
- **Dirtree/\$ dirtree/bookmarks_enumplace_t**
- **Dirtree/\$ dirtree/bpts**

They deal with merging of the standard dirtrees.

The "Detail" pane is absent.

File Edit Jump Search Windows Help

Progress

Step: idaplace_t
 Dirtree: tinfos
 Conflicts: 0
 Line 66 of 117

Local		Remote	
inode	Directory	inode	Directory
CPPEH_RECORD	/	CPPEH_RECORD	/
EH3_EXCEPTION_REGISTRATION	/	EH3_EXCEPTION_REGISTRATION	/
PEH3_EXCEPTION_REGISTRATION	/	PEH3_EXCEPTION_REGISTRATION	/
PSCOPETABLE_ENTRY	/	PSCOPETABLE_ENTRY	/
._EH3_SCOPETABLE	/	._EH3_SCOPETABLE	/
._EH4_SCOPETABLE	/	._EH4_SCOPETABLE	/
EH4_SCOPETABLE_RECORD	/pdb	EH4_SCOPETABLE_RECORD	/

Line 7 of 7

File Edit Jump Search Windows Help

Progress

Step: Dirtree: tinfos
 structs
 Conflicts: 0
 Line 67 of 117

Local		Remote	
inode	Directory	inode	Directory
CPPEH_RECORD	/	CPPEH_RECORD	/rsys
._EH3_EXCEPTION_REGISTRATION	/	._EH3_EXCEPTION_REGISTRATION	/rsys
._EH4_SCOPETABLE	/	._EH4_SCOPETABLE	/rsys/EH4
EH4_SCOPETABLE_RECORD	/	EH4_SCOPETABLE_RECORD	/rsys/EH4

Line 2 of 4

3.3.29. Misc/Try blocks

Merging of try and catch block info.

The "Detail" pane describes try block.

The screenshot displays a debugger window with the following components:

- Menu Bar:** File, Edit, Jump, Search, Windows, Help.
- Progress Bar:** Progress
- Step/Conflicts:** Step: bpts, Conflicts: 0. A tree view shows 'Misc' expanded to 'Try blocks' with a count of 1.
- Line 82 of 120:** A header for the current step.
- Local View:** A table with columns 'Base address' and 'Qty'. One row is highlighted:

Base address	Qty
.text:0000044	2
- Remote View:** A table with columns 'Base address' and 'Qty'. Two rows are highlighted:

Base address	Qty
.text:000017C	2
.text:00004B7	2
- Detail Pane (Left):** Shows the structure for '.text:0000044, 2':


```

      ranges: 1
      {
      .text:0000044...text:000021B
      }
      kind: MS SEH,
      nesting level: 0
      ranges: 1
      {
      .text:0000044...text:000021B
      }
      kind: C++ try/catch,
      nesting level: 0
      
```
- Detail Pane (Right):** Shows the structure for '.text:000017C, 2':

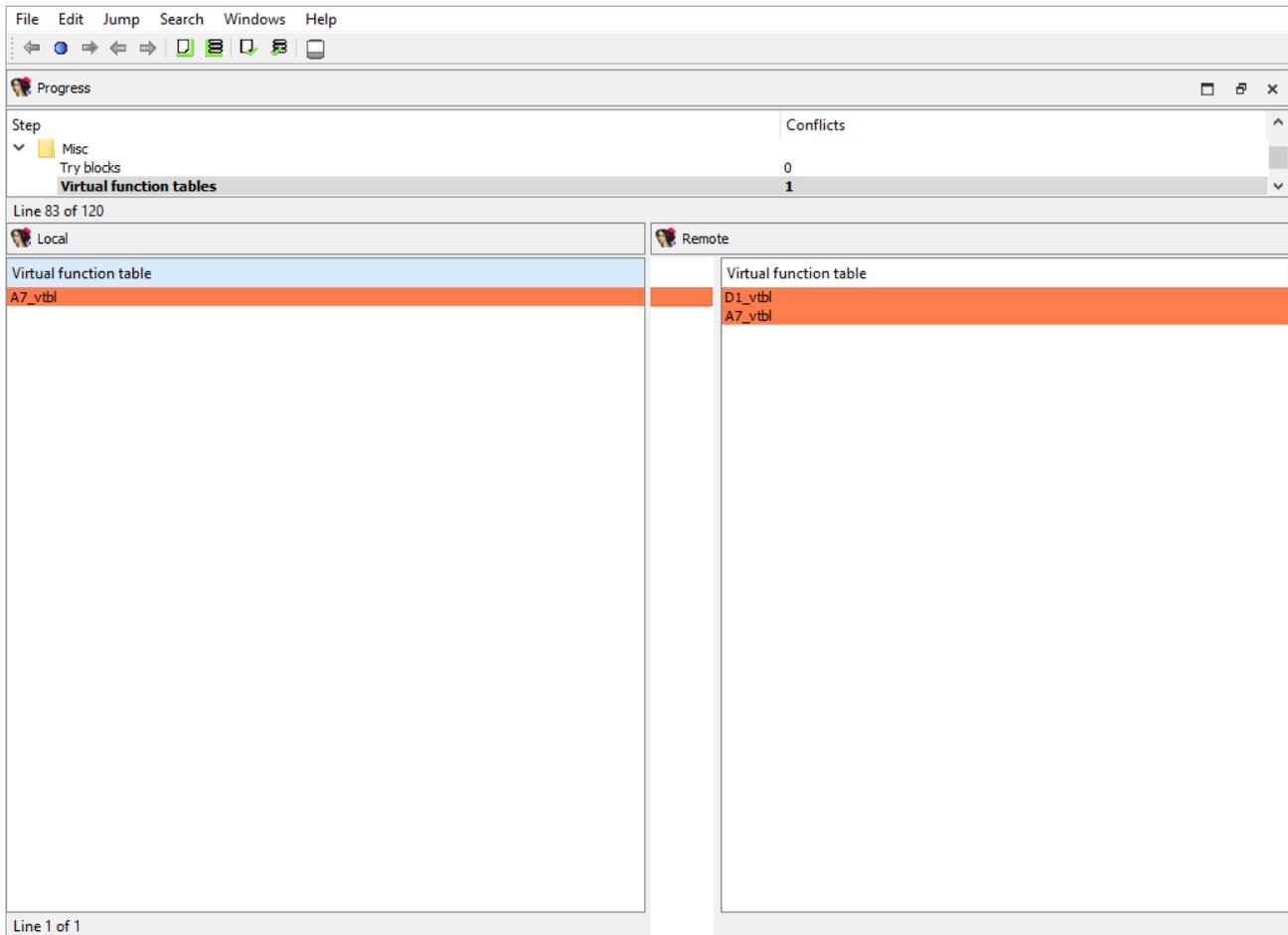

```

      ranges: 1
      {
      .text:000017C...text:0000218
      }
      kind: MS SEH,
      nesting level: 0
      ranges: 1
      {
      .text:000017C...text:0000218
      }
      kind: C++ try/catch,
      nesting level: 0
      > .text:00004B7, 2
      
```

3.3.30. Misc/Virtual function tables

Merging of virtual function tables.

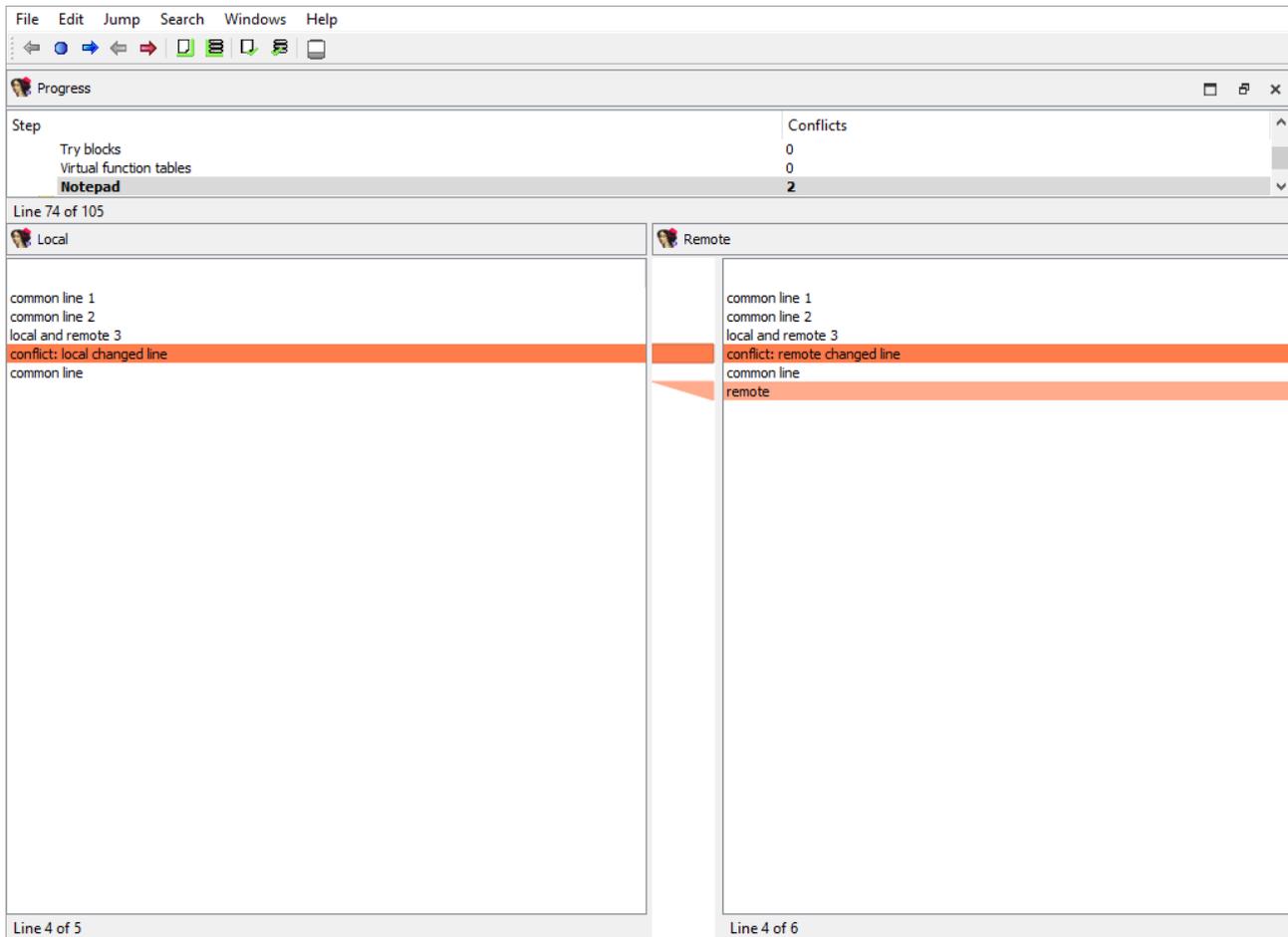
The "Detail" pane is absent.



3.3.31. Misc/Notepad

Merging of database notepads. Each line of text is a *diffpos*.

The "Detail" pane is absent.



3.3.32. Processor specific/...

Each processor plugin creates its own merge steps to handle the processor plugin's specific data.

For example, the PC processor module adds the following merge steps:

- Processor specific/Analyze ea for a possible offset
- Processor specific/Frame pointer info
- Processor specific/Pushinfo
- Processor specific/VXD info 2
- Processor specific/Callee EA|AH value
- ...

File Edit Jump Search Windows Help

Progress

Step	Conflicts
Frame info	0
SEH block	0
Pushinfo	1

Line 99 of 120

Local	Remote
.init_proc flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	.init_proc flags: HAVE_SSIZE PSI_FLAGS
sub_8048378 flags: HAVE_SSIZE PSI_FLAGS spoiled: 00	sub_8048378 flags: HAVE_SSIZE PSI_FLAGS
__gmon_start__ flags: HAVE_SSIZE PSI_FLAGS spoiled: 00	__gmon_start__ flags: HAVE_SSIZE PSI_FLAGS
__libc_start_main flags: HAVE_SSIZE PSI_FLAGS spoiled: 00	__libc_start_main flags: HAVE_SSIZE PSI_FLAGS
_start flags: HAVE_SSIZE PSI_FLAGS spoiled: 00	_start flags: HAVE_SSIZE PSI_FLAGS
__do_global_ctors_aux flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	__do_global_ctors_aux flags: HAVE_SSIZE PSI_FLAGS
frame_dummy flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	frame_dummy flags: HAVE_SSIZE PSI_FLAGS
_Z10return_intPc flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	_Z10return_intPc flags: HAVE_SSIZE PSI_FLAGS
main flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	main flags: HAVE_SSIZE PSI_FLAGS
__libc_csu_fini flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	__libc_csu_fini flags: HAVE_SSIZE PSI_FLAGS
__libc_csu_init flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	__libc_csu_init flags: HAVE_SSIZE PSI_FLAGS
__i686.get_pc_thunk.bx flags: HAVE_SSIZE PSI_FLAGS spoiled: 00	__i686.get_pc_thunk.bx flags: HAVE_SSIZE PSI_FLAGS
__do_global_ctors_aux flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	__do_global_ctors_aux flags: HAVE_SSIZE PSI_FLAGS
.term_proc flags: HAVE_SSIZE PSI_FLAGS prolog: [80]	.term_proc flags: HAVE_SSIZE PSI_FLAGS

Line 8 of 14

Local: **_Z10return_intPc**
 flags: HAVE_SSIZE PSI_FLAGS
 prolog: [8048475]
 pops: [8048490]
 psi(ea:off:width:reg:flags): 8048474:0:4:bp:0 8048477:4:4:STACK:0
 bpidix: 1
 spoiled: 00000034

Remote: **_Z10return_intPc**
 flags: HAVE_SSIZE PSI_FLAGS
 prolog: [8048475]
 pops: [8048490]
 psi(ea:off:width:reg:flags): 8048474:0:4:bp:0 8048477:4:4:STACK:0
 bpidix: 1
 spoiled: 00000035

File Edit Jump Search Windows Help

Progress

Step	Conflicts
Pushinfo	0
VXD info 1	0
VXD info 2	1

Line 101 of 120

Local	Remote
18 ExQueueWorkItem spoff: 8 comment: (inplace)	18 ExQueueWorkItem spoff: 8 comment: remote_cm

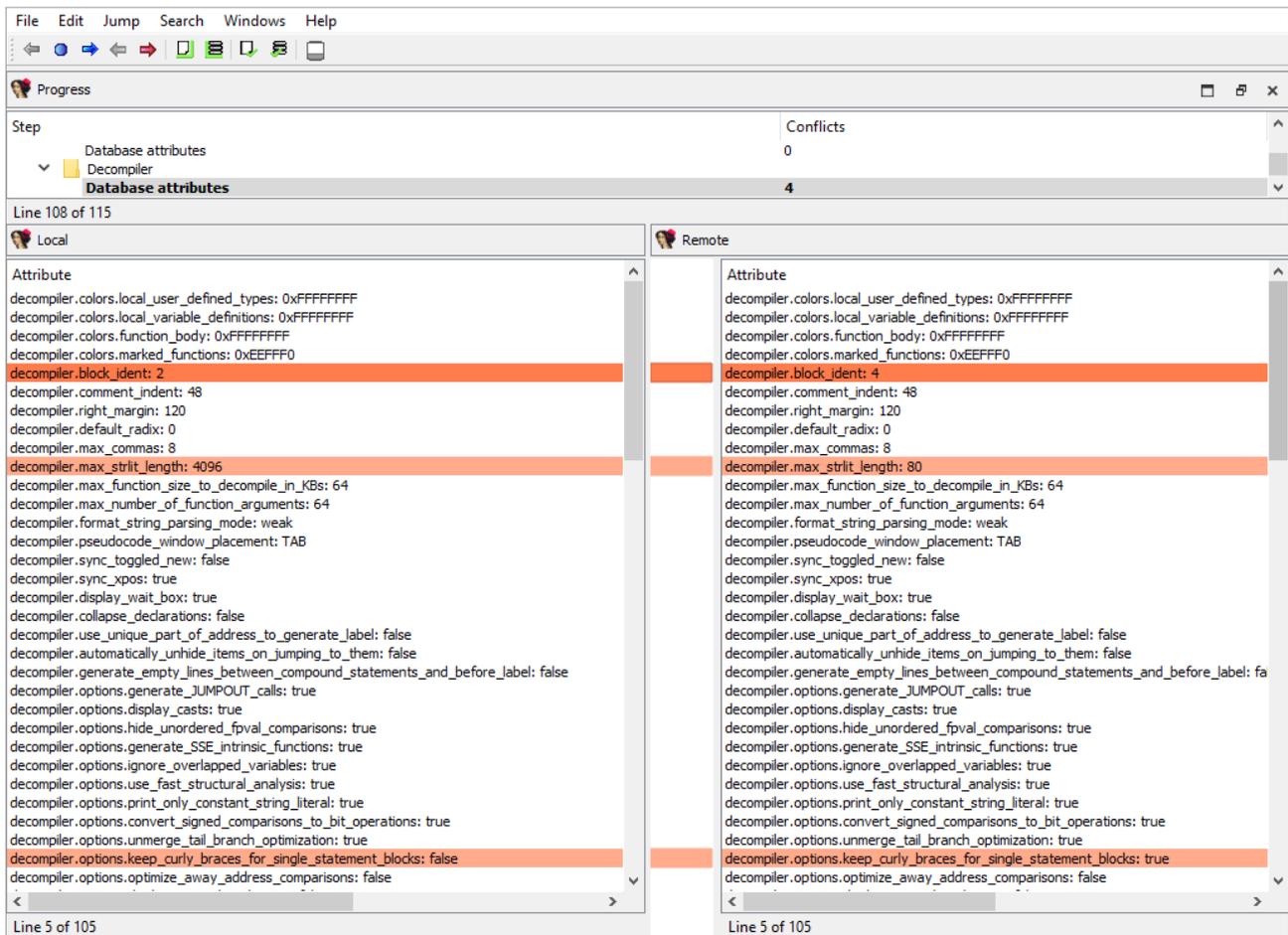
Line 1 of 1

Local: **18 ExQueueWorkItem**
 spoff: 8
 comment: (inplace)

Remote: **18 ExQueueWorkItem**
 spoff: 8
 comment: remote_cm

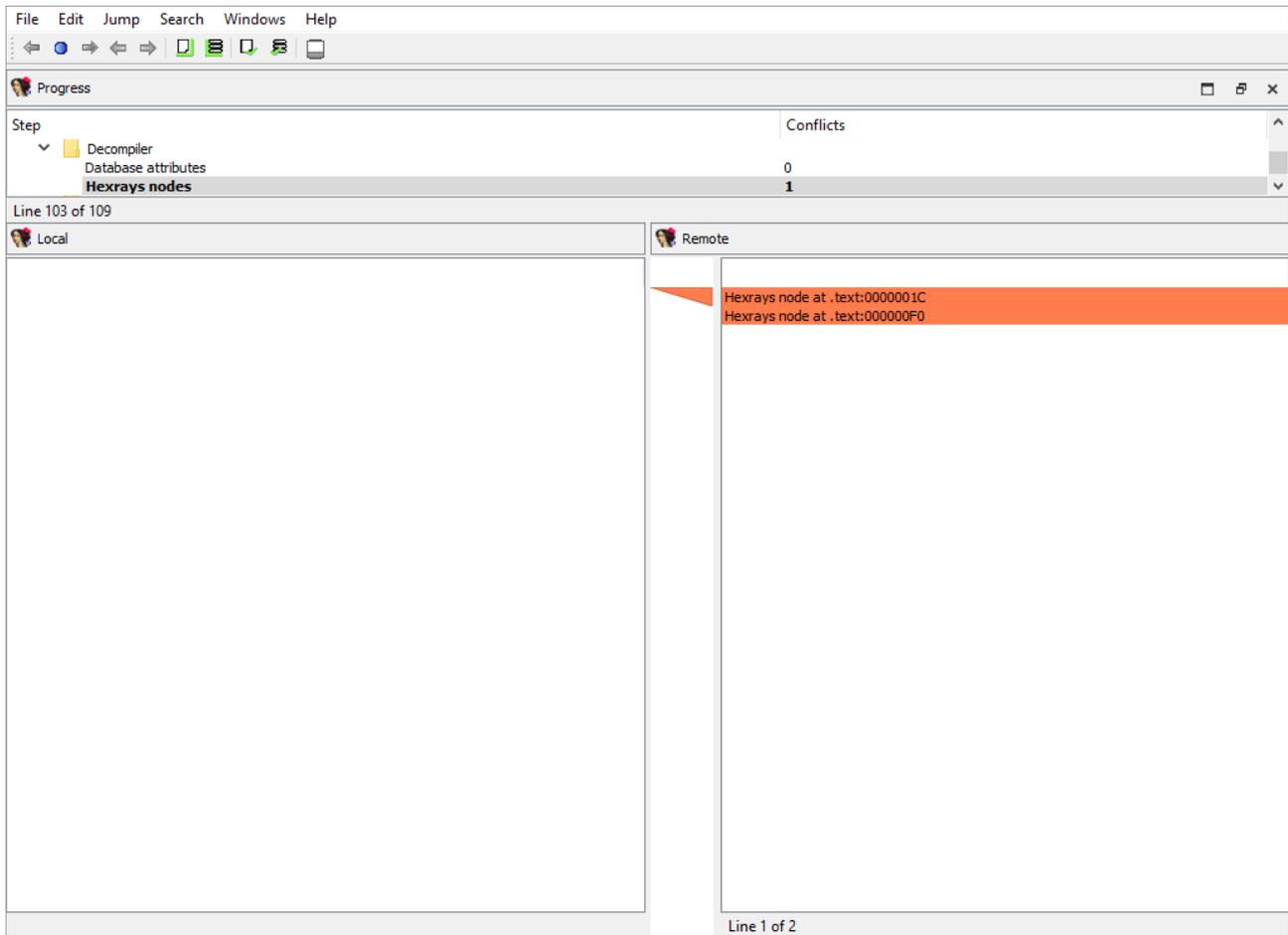
3.3.33. Plugins/Decompiler/...

Merging of the decompiler data starts with the global configuration parameters from hexrays.cfg:



To handle decompilation of specific functions, IDA stores the decompilation data in a database netnode named **Hexrays node**.

The merge step **Plugins/Decompiler/Hexrays nodes** adds or deletes netnodes, indicating which functions have or haven't been decompiled in each databases:



The decompilation data for matching functions is compared using the following attributes:

- **Plugins/Decompiler/.../Numforms**
- **Plugins/Decompiler/.../mflags**
- **Plugins/Decompiler/.../User-defined funcargs**
- **Plugins/Decompiler/.../User-defined variable mapping**
- **Plugins/Decompiler/.../User-defined lvar info**
- **Plugins/Decompiler/.../lvar settings**
- **Plugins/Decompiler/.../IFLAGS**
- **Plugins/Decompiler/.../User labels**
- **Plugins/Decompiler/.../User unions**
- **Plugins/Decompiler/.../User comments**
- **Plugins/Decompiler/.../User-defined call**

If there is a difference, each comparison criteria will be assigned its own merge step. Each step will use the standard "Pseudocode" widget so that differences can be viewed in-context with the full pseudocode:

File Edit Jump Search Windows Help

Progress

Step: Decompiler, quotearg_n_style_colon at 13360, Numforms, 1

Line 109 of 133

Local	Remote
<pre> WORDS(v14[0]) = v8; WORD6(v14[0]) = v9; HIWORD(v14[0]) = v10; v14[1] = v11; v14[2] = v12; *(__QWORD *)&v14[3] = v13; set_char_quoting(v14, 58LL, 1LL); return ((__int64 (*) (__QWORD, unsigned __int64, __QWORD, ...))quotearg) </pre>	<pre> HIWORD(v18[3]) = v12; LODWORD(v18[4]) = v13; HIWORD(v18[4]) = v14; LODWORD(v18[5]) = v15; HIWORD(v18[5]) = v16; v18[6] = v17; set_char_quoting(v18, 58LL, 1LL); return ((__int64 (*) (__QWORD, unsigned __int64, __QWORD, ...))) </pre>
<p>00013360 quotearg_n_style_colon:37 (13360)</p> <p>133D0, opnum=1 HEX, radix 16 SIGNED FIXED</p>	<p>00013360 quotearg_n_style_colon:45 (13360)</p> <p>133D0, opnum=1 ENUM enum_1 FIXED VALID</p>

File Edit Jump Search Windows Help

Progress

Step: quotearg_n_style_colon at 13360, Numforms, mflags, 0

Line 110 of 133

Local	Remote
<pre> LOWORD(v14[0]) = v3; WORD1(v14[0]) = v4; WORD2(v14[0]) = v5; WORD3(v14[0]) = v6; WORD4(v14[0]) = v7; WORDS(v14[0]) = v8; WORD6(v14[0]) = v9; HIWORD(v14[0]) = ; v14[1] = v11; v14[2] = v12; *(__QWORD *)&v14[3] = v13; set_char_quoting(v14, 58LL, 1LL); return ((__int64 (*) (__QWORD, unsigned __int64, __QWORD, ...))quotearg) </pre>	<pre> __int64 v17; // [xsp+60h] [xbp+60h] __int64 v18[7]; // [xsp+70h] [xbp+70h] BYREF __int64 v20; // x4 quoting_options_from_style(a2, 0LL); v18[0] = v3; v18[1] = v4; LOWORD(v18[2]) = v5; WORD1(v18[2]) = v6; WORD2(v18[2]) = v7; HIWORD(v18[2]) = ; LOWORD(v18[3]) = v9; WORD1(v18[3]) = v10; WORD2(v18[3]) = v11; HIWORD(v18[3]) = v12; </pre>
<p>00013360 quotearg_n_style_colon:32 (13360)</p> <p>133B8, opnum=0 MAXCMB2</p>	<p>00013360 quotearg_n_style_colon:32 (13360)</p> <p>133BC, opnum=0 MAXCMB2 133C0, opnum=0 MAXCMB4</p>

File Edit Jump Search Windows Help

Progress

Step	Conflicts
mflags	0
User-defined funcargs	0
User-defined variable mapping	1

Line 111 of 133

Local

```

WORDS(v14[0]) = v8;
WORDE(v14[0]) = v9;
HIWORD(v14[0]) = v10;
v14[1] = v11;
v14[2] = v12;
*(__QWORD *)&v14[3] = v13;
set_char_quoting(v14, 58LL, 1LL);
return ((__int64 (__QWORD, unsigned __int64, __QWORD, ...))quotearg
)

```

00013360 quotearg_n_style_colon:37 (13360)

call_ea: 133DC

- __QWORD
- dev_t
- __QWORD
- __QWORD

Remote

```

HIWORD(v18[3]) = v12;
LODWORD(v18[4]) = v13;
HIDWORD(v18[4]) = v14;
LODWORD(v18[5]) = v15;
HIDWORD(v18[5]) = v16;
v18[6] = v17;
set_char_quoting(v18, 58LL, 1LL);
return ((__int64 (__QWORD, unsigned __int64, __QWORD, ...))
)

```

00013360 quotearg_n_style_colon:45 (13360)

call_ea: 133DC

- __QWORD
- unsigned __int64
- __QWORD
- __QWORD
- __QWORD

File Edit Jump Search Windows Help

Progress

Step	Conflicts
mflags	0
User-defined funcargs	0
User-defined variable mapping	1

Line 112 of 129

Local

```

// Comments for function quotearg_n_style_colon:
// Line 1 (remote)
// Line 2 (common)
// Line 3 (remote)
// local variable allocation has failed, the output may be wrong
__int64 __fastcall quotearg_n_style_colon(
    unsigned int a1,
    unsigned int a2,
    dev_t a3)
{
    __int6 v3; // [xsp+30h] [xbp+30h] lvar comment for v6 (local)
    __int6 v4; // [xsp+32h] [xbp+32h]
    __int6 v5; // [xsp+34h] [xbp+34h]
    __int6 v6; // [xsp+36h] [xbp+36h]
    __int6 v7; // [xsp+38h] [xbp+38h]
}

```

00013360 quotearg_n_style_colon:1 (13360)

Remote

```

// Comments for function quotearg_n_style_colon:
// Line 1 (remote)
// Line 2 (common)
// Line 3 (remote)
__int64 __fastcall quotearg_n_style_colon(
    unsigned int a1,
    unsigned int a2,
    unsigned __int64 a3)
{
    __int64 v3; // [xsp+30h] [xbp+30h] lvar comment for v6 (remo
    __int64 v4; // [xsp+32h] [xbp+32h]
    __int6 v5; // [xsp+34h] [xbp+34h]
    __int6 v6; // [xsp+36h] [xbp+36h]
    __int6 v7; // [xsp+38h] [xbp+38h]
    __int6 v8; // [xsp+40h] [xbp+40h]
    __int6 v9; // [xsp+42h] [xbp+42h]
    __int6 v10; // [xsp+44h] [xbp+44h]
    __int6 v11; // [xsp+46h] [xbp+46h]
}

```

00013360 quotearg_n_style_colon:6 (13360)

unsigned int a1; // w0 ISARG MAPDST

File Edit Jump Search Windows Help

Progress

Step	Conflicts
User-defined funcargs	0
User-defined variable mapping	0
User-defined lvar info	1

Line 113 of 133

Local

```

__int16 v3; // [xsp+30h] [xbp+30h]
__int128 v14[4]; // [xsp+70h] [xbp+70h] OVERLAPPED BYREF
dev_t a3; // x2 ISARG MAPDST
        
```

Line 2 of 3

- ▾ `__int16 v3; // [xsp+30h] [xbp+30h]`
 - `cmt: lvar comment for v6 (local)`
- > `__int128 v14[4]; // [xsp+70h] [xbp+70h] OVERLAPPED BYREF`
- > `dev_t a3; // x2 ISARG MAPDST`

Remote

```

__int64 v3; // [xsp+30h] [xbp+30h]
__int64 v18[7]; // [xsp+70h] [xbp+70h] BYREF
unsigned __int64 a3; // x2 ISARG MAPDST
        
```

Line 2 of 3

- ▾ `__int64 v3; // [xsp+30h] [xbp+30h]`
 - `cmt: lvar comment for v6 (remote)`
 - `type: __int64`
- > `__int64 v18[7]; // [xsp+70h] [xbp+70h] BYREF`
- > `unsigned __int64 a3; // x2 ISARG MAPDST`

File Edit Jump Search Windows Help

Progress

Step	Conflicts
User-defined variable mapping	0
User-defined lvar info	0
lvar settings	1

Line 125 of 134

Local

Remote

```

stkoff_delta: 0x0
ulv_flags:
        
```

Line 1 of 2

File Edit Jump Search Windows Help

Progress

Step: lvar settings
save_abbr at 15448
IFLAGS

Conflicts: 0

Line 116 of 133

Address	Op	Collapsed
154A4	if	Yes
154A4	while	Yes

Local Remote

Address

Line 1 of 2

File Edit Jump Search Windows Help

Progress

Step: save_abbr at 15448
IFLAGS
User labels

Conflicts: 0

Line 117 of 133

Local	Remote
<pre> u3_t *v8; // x20 char *c7; // x21 signed __int64 v10; // x21 const char *v11; // x22 __int64 v13; // x24 v11 = *(const char **)(a2 + 48); if (!v11) return ILL; v8 = a1; if (a2 <= (unsigned __int64)v11 && (unsigned __int64)v11 < a2 + 56) return ILL; v7 = &a1->cl6[9]; if (!*v11) { v7 = ""; goto LOCAL_13; } CONFLICT_LOCAL_5: c7 = v8->s88.u8.cc7.c7; if (!strcmp(v7, v11)) goto LOCAL_13; while (1) { if (*v7) goto COMMON_10; if (c7 != v7) break; if (!v8->s88.u8.c8[0]) { v13 = strlen(v11) + 1LL; v10 = 0LL; goto LABEL_21; } } COMMON_10: v7 += add_one(strlen(v7)); 00015448 save_abbr:27 (15448) </pre>	<pre> u3_t *v8; // x20 char *v9; // x21 signed __int64 v10; // x21 const char *v11; // x22 __int64 v13; // x24 v11 = *(const char **)(a2 + 48); if (!v11) return ILL; v8 = a1; if (a2 <= (unsigned __int64)v11 && (unsigned __int64)v11 < a2 + 56) return ILL; c7 = a1->s88.u8.cc7.c7; if (!*v11) { c7 = ""; goto LABEL_13; } CONFLICT_REMOTE_5: v9 = v8->cc7[11].c7; if (!strcmp(c7, v11)) goto LABEL_13; while (1) { if (*c7) goto COMMON_10; if (v9 != c7) break; if (!v8->s88.u8.c8[0]) { v13 = strlen(v11) + 1LL; v10 = 0LL; goto REMOTE_21; } } COMMON_10: c7 += inc(strlen(c7)); 00015448 save_abbr:27 (15448) </pre>

File Edit Jump Search Windows Help

Progress

Step	Conflicts
IFLAGS	0
User labels	0
User unions	2

Line 118 of 133

Local

```

v11 = *(const char **) (a2 + 48);
if ( !v11 )
    return ILL;
v8 = a1;
if ( a2 <= (unsigned __int64)v11 && (unsigned __int64)v11 < a2 + 56
    return ILL;
v7 = &a1->c16[9];
if ( !*v11 )
{
    v7 = "";
    goto LOCAL_13;
}
CONFLICT_LOCAL_5:
v7 = v7->s88.u8.cc7.c7;
00015448 save_abbr:21 (15448)
        
```

15488 fields: 1

15490 fields: 0,1

Remote

```

v11 = *(const char **) (a2 + 48);
if ( !v11 )
    return ILL;
v8 = a1;
if ( a2 <= (unsigned __int64)v11 && (unsigned __int64)v11 <
    return ILL;
v7 = a1->s88.u8.cc7.c7;
if ( !*v11 )
{
    c7 = "";
    goto LABEL_13;
}
CONFLICT_REMOTE_5:
v7 = v7->cc7[1].c7;
00015448 save_abbr:21 (15448)
        
```

15488 fields: 0,1

15490 fields: 2

File Edit Jump Search Windows Help

Progress

Step	Conflicts
lvar settings	0
hash_pjw at 15A38	0
User comments	1

Line 126 of 139

Local

```

unsigned __int8 v4; // w3
int v5; // t1

v4 = *a1;
if ( !*a1 )
    return OLL;
collision_name = OLL;
// Block comment for 'do' statementent
// common line 1
// local comment
// common line 2
// extra line (local)
do
{
    collision_name = __ROR8__(collision_name, 55) + v4;
00015A38 hash_pjw:13 (15A38)
        
```

15A54 /*pre*/
Block comment for 'do' statementent
common line 1
local comment
common line 2
extra line (local)

15A60;

Remote

```

unsigned __int8 collision_name; // w3
int v5; // t1

collision_name = *a1;
if ( !*a1 )
    return OLL;
v3 = OLL;
// Block comment for 'do' statementent
// common line 1
// remote comment
// common line 2
do
{
    v3 = __ROR8__(v3, 55) + collision_name;
    v5 = *++a1;
00015A38 hash_pjw:13 (15A38)
        
```

15A54 /*pre*/
Block comment for 'do' statementent
common line 1
remote comment
common line 2

15A60;

The screenshot displays a debugger's 'Progress' window with a table of steps:

Step	Conflicts
mflags	0
User-defined funcargs	0
User-defined call	1

Below the table, the debugger shows 'Line 121 of 133' and two side-by-side code windows: 'Local' and 'Remote'.

Local View:

```

if ( !v8->s88.u8.c8[0] )
{
    v13 = strlen(v11) + 1LL;
    v10 = 0LL;
    goto LABEL_21;
}
COMMON_10:
v7 += add_one(strlen(v7));
if ( *v7 )
    goto CONFLICT_LOCAL_5;
i8 = v8->s88.i8;
if ( !v8->s88.i8 )
    goto CONFLICT_LOCAL_5;
LWORD(v8) = v8->s88.i8;
WORD1(v8) = WORD1(i8);
00015448 save_abbr:44 (15448)

```

Remote View:

```

if ( !v8->s88.u8.c8[0] )
{
    v13 = strlen(v11) + 1LL;
    v10 = 0LL;
    goto REMOTE_21;
}
COMMON_10:
v7 += inc(strlen(v7));
if ( *v7 )
    goto CONFLICT_REMOTE_5;
i8 = (u3_t *)v8->s88.i8;
if ( !v8->s88.i8 )
    goto CONFLICT_REMOTE_5;
LWORD(v8) = v8->s88.i8;
HIDWORD(v8) = HIDWORD(i8);
00015448 save_abbr:44 (15448)

```

At the bottom, the debugger shows the function signatures for the calls:

- Local:** 154C4 add_one
`unsigned __int64 __fastcall add_one(__int64)`
- Remote:** 154C4 inc
`__int64 __fastcall inc(__int64 __OP1)`

3.3.34. Loader data merge phases

The file loader that was used to create the database may have stored some data in the database that is specific to the loader itself.

There are merge phases for each loader, for example:

- Loader/PE file/...
- Loader/NE file/...
- Loader/ELF file/...
- Loader/TLS/...
- Loader/ARM segment flags/...

File Edit Jump Search Windows Help

Progress

Step Loader Conflicts

PE file Database attributes 2

Line 77 of 116

Local	Remote
Attribute	Attribute
penode.PE_header:	penode.PE_header: 504500004C010100E86AA638C66F82E9CC0E82F878000F030B0131316
penode.dbginfo_fpos: 0	penode.dbginfo_fpos: 0
penode.loading_address: 0	penode.loading_address: 0x400000
penode.PE_header_offset: 0	penode.PE_header_offset: 0xC
penode.NE_flags: 0	penode.NE_flags: 0xA5
penode.TDS_loaded: 0	penode.TDS_loaded: 0
penode.POSIX_x86_imported: 0	penode.POSIX_x86_imported: 0
penode.overlay_RVA: 0	penode.overlay_RVA: 0
penode.overlay_size: 0	penode.overlay_size: 0
penode.PDB_filename:	penode.PDB_filename:
penode.use_Native_API: 0	penode.use_Native_API: 0
penode.relocation_info:	penode.relocation_info:
penode.RSDS:	penode.RSDS:
penode.NB10:	penode.NB10:
penode.UADS:	penode.UADS:

Line 1 of 15

File Edit Jump Search Windows Help

Progress

Step elf_segment_headers Conflicts

ARM segment flags arm_segment_flags 1

Line 78 of 108

Local	Remote
COMMON ALIGN=2	

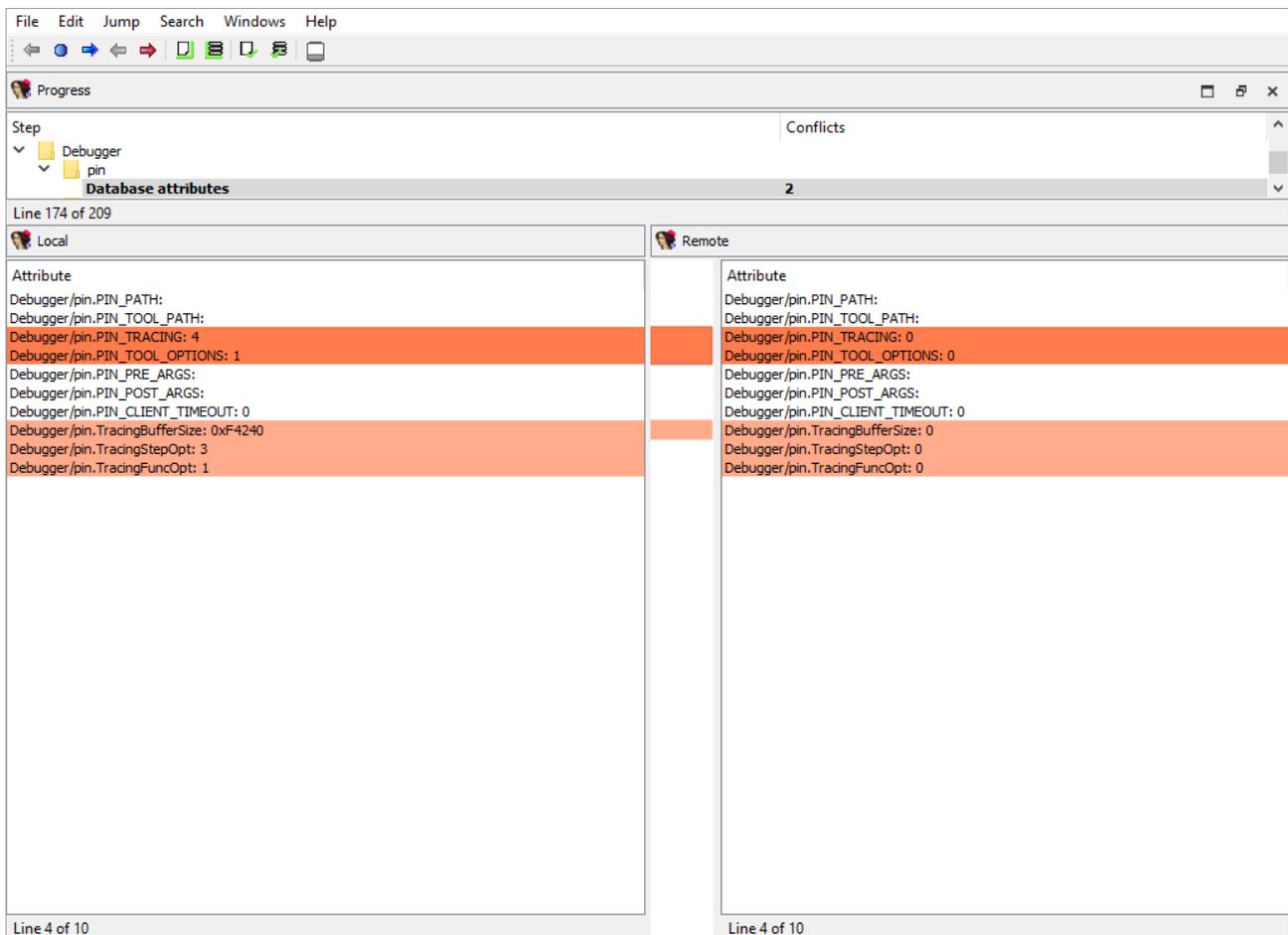
Line 1 of 1

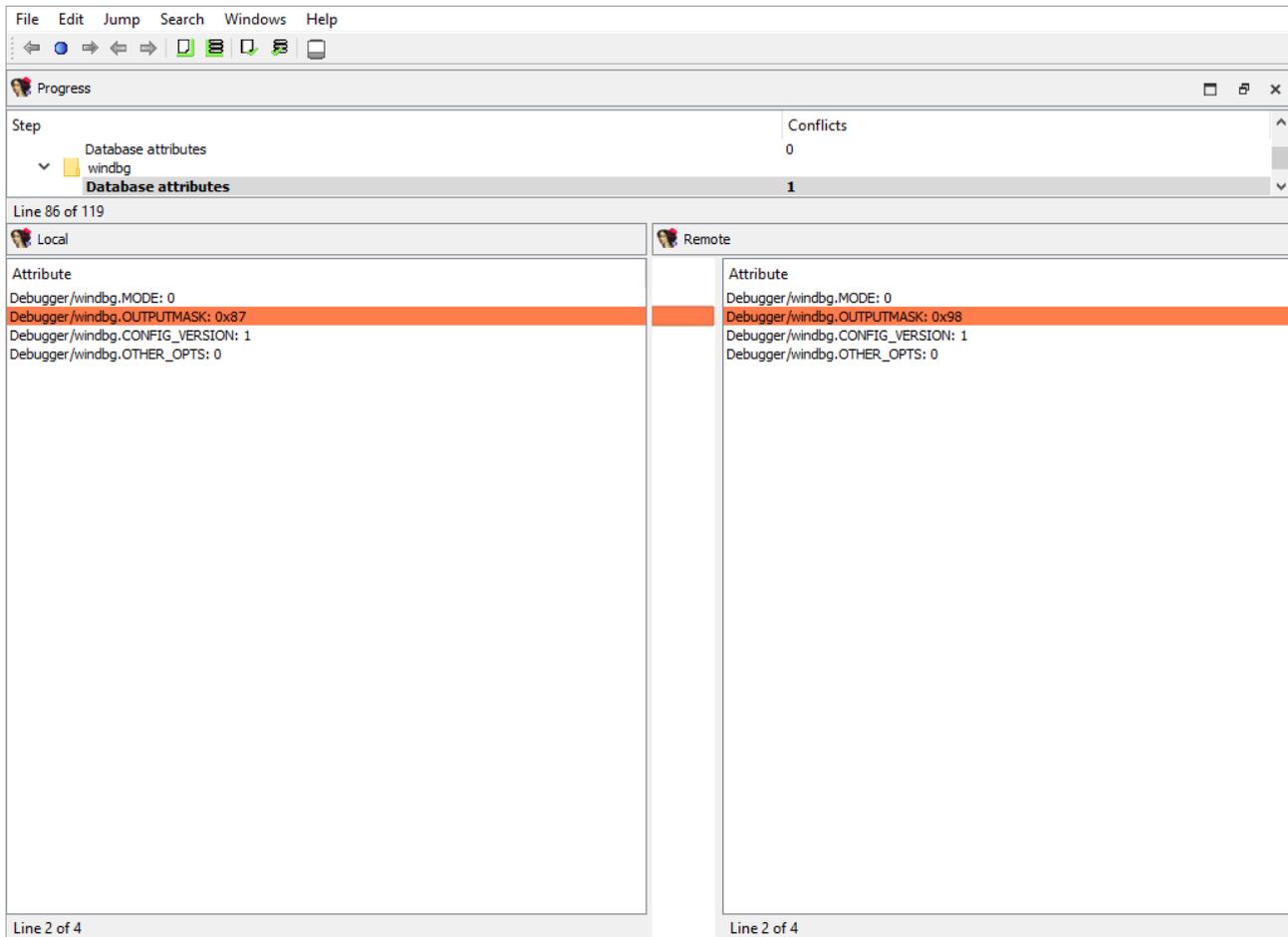
3.3.35. Debugger data merge phases

To handle the differences in debugger data the following merge steps may be created:

- **Debugger/pin**
- **Debugger/gdb**
- **Debugger/xnu**
- **Debugger/ios**
- **Debugger/bochs**
- **Debugger/windbg**
- **Debugger/rmac_arm**
- **Debugger/lmac_arm**
- **Debugger/rmac**
- **Debugger/lmac**

As can be deduced by their names, they handle debugger-specific data in the database.





3.3.36. Other plugins merge phases

There are a number of IDA plugins that need to merge their data.

For example:

- **Plugins/PDB**
- **Plugins/golang**
- **Plugins/EH_PARSE**
- **Plugins/Callgraph**
- **Plugins/swift**

Any third party plugin may add merge phases using the IDA SDK. We provide sample plugins that illustrate how to add support for merging into third party plugins.

File Edit Jump Search Windows Help

Progress

Step Database attributes Conflicts 0

PDB Database attributes 1

Line 102 of 115

Local	Remote
Attribute	Attribute
pdb.loading_result: 0	pdb.loading_result: 1

File Edit Jump Search Windows Help

Progress

Step Plugins Conflicts 2

swift Database attributes

Line 121 of 134

Local	Remote
Attribute	Attribute
swift.demangler_enabled: true	swift.demangler_enabled: false
swift.detection_allowed: false	swift.detection_allowed: false
swift.swift_version: 6	swift.swift_version: 6
swift.path_to_demangle_library:	swift.path_to_demangle_library: Z:\\dasrc\\current\\bin\\x64_win_vc_opt\\libSwiftDemangle.

Line 1 of 4

4. Appendix B

4.1. Using IDASDK to add merge functionality to plugin

4.1.1. Overview

Any plugin that stores its data in the database must implement the logic for merging its data. For that, the plugin must provide the description of its data and ask the kernel to create merge handlers based on these descriptions.

The kernel will use the created handlers to perform merging and to display merged data to the users. The plugin can implement callback functions to modify some aspects of merging, if necessary.

The plugin may have two kinds of data with permanent storage:

1. Data that applies to entire database (e.g. the options). To describe this data, the `idbattr_info_t` type is used.
2. Data that is tied to a particular address. To describe this data, the `merge_node_info_t` type is used.

The kernel will notify the plugin using the `processor_t::ev_create_merge_handlers` event. On receiving it, the plugin should create the merge handlers, usually by calling the `create_merge_handlers()` function.

4.1.2. Plugin

The IDA SDK provides several sample plugins to demonstrate how to add merge functionality to third party plugins:

- mex1/
- mex2/
- mex3/
- mex4/

The sample plugin without the merge functionality consists of two files:

- mex.hpp
- mex_impl.cpp

It is a regular implementation of a plugin that stores some data in the database. Please check the source files for more info.

We demonstrate several approaches to add the merge functionality. They are implemented in different directories mex1/, mex2/, and so on.

The `MEX_N` macros that are defined in makefile are used to parameterize the plugin implementation, so that all plugin examples may be used simultaneously.

You may check the merge results for the plugins in one session of IDA Teams. Naturally, you should prepare databases by running plugins before launching of IDA Teams session.

4.1.3. Merge functionality

The merge functionality is implemented in the `merge.cpp` file. It contains `create_merge_handlers()`, which is responsible for the creation of the merge handlers.

Variants:

mex1/

Merge values are stored in netnodes. The kernel will read the values directly from netnodes, merge them, and write back. No further actions are required from the plugin. If the data is stored in a simple way using altvals or supvals, this simple approach is recommended.

mex2/

Merge values are stored in variables (in the memory). For more complex data that is not stored in a simple way in netnodes, (for example, data that uses database blobs), the previous approach cannot be used. This example shows

how to merge the data that is stored in variables, like fields of the plugin context structure. The plugin provides the field descriptions to the kernel, which will use them to merge the data in the memory. After merging, the plugin must save the merged data to the database.

mex3/

Uses mex1 example and illustrates how to improve the UI look.

mex4/

Merge data that is stored in a netnode blob. Usually blob data is displayed as a sequence of hexadecimal digits in a merge chooser column. We show how to display blob contents in detail pane.

5. Resolving conflicts in a file

When a user needs to commit changes made to a file, but that same file has received other modifications (likely from other users) in the meantime, it is necessary to first "merge" the two sets of modifications together.

When the two sets of modifications do not overlap, merging is trivial - at least conceptually. But when they do overlap, they produce conflict(s).

Since IDA Teams focuses on collaboration over IDA database files, the rest of this section will focus on the different strategies that are available for resolving conflicts among those.

IDA Teams comes with multiple strategies to help in conflict resolution of IDA database files:

- [Auto-resolve \(if no conflicts\)](#)
- [Auto-resolve, prefer local](#)
- [Auto-resolve, prefer remote](#)
- [Interactive merge mode](#)
- [Use local, discard remote](#)
- [Use remote, discard local](#)

5.1. Auto-resolve (if no conflicts)

Launch IDA in a non-interactive batch mode, attempting to perform all merging automatically.

If any conflict is discovered, bail out of the merge process, and don't modify the local database.

5.2. Auto-resolve, prefer local

Launch IDA in a non-interactive batch mode, attempting to perform all merging automatically.

If a conflict is discovered, assume that the "local" change (i.e., the current user's change) is the correct one, and apply that.

Once all merging is done and conflicts are resolved, write those to the local database and exit IDA

5.3. Auto-resolve, prefer remote

Launch IDA in a non-interactive batch mode, attempting to perform all merging automatically.

If a conflict is discovered, assume that the "remote" change (i.e., the change made by another user) is the correct one, and apply that.

Once all merging is done and conflicts are resolved, write those to the local database and exit IDA

5.4. Interactive merge mode

Manual merge mode.

This will launch IDA in an interactive, 3-pane mode, allowing the user to decide how to resolve each conflict.

Once all merging is done and conflicts are resolved, exit IDA and write the changes to the local database.

5.5. Use local, discard remote

Select the local database, ignoring all changes in the remote database.

No IDA process is run.

5.6. Use remote, discard local

Select the remote database, ignoring all changes in the local database.

No IDA process is run.