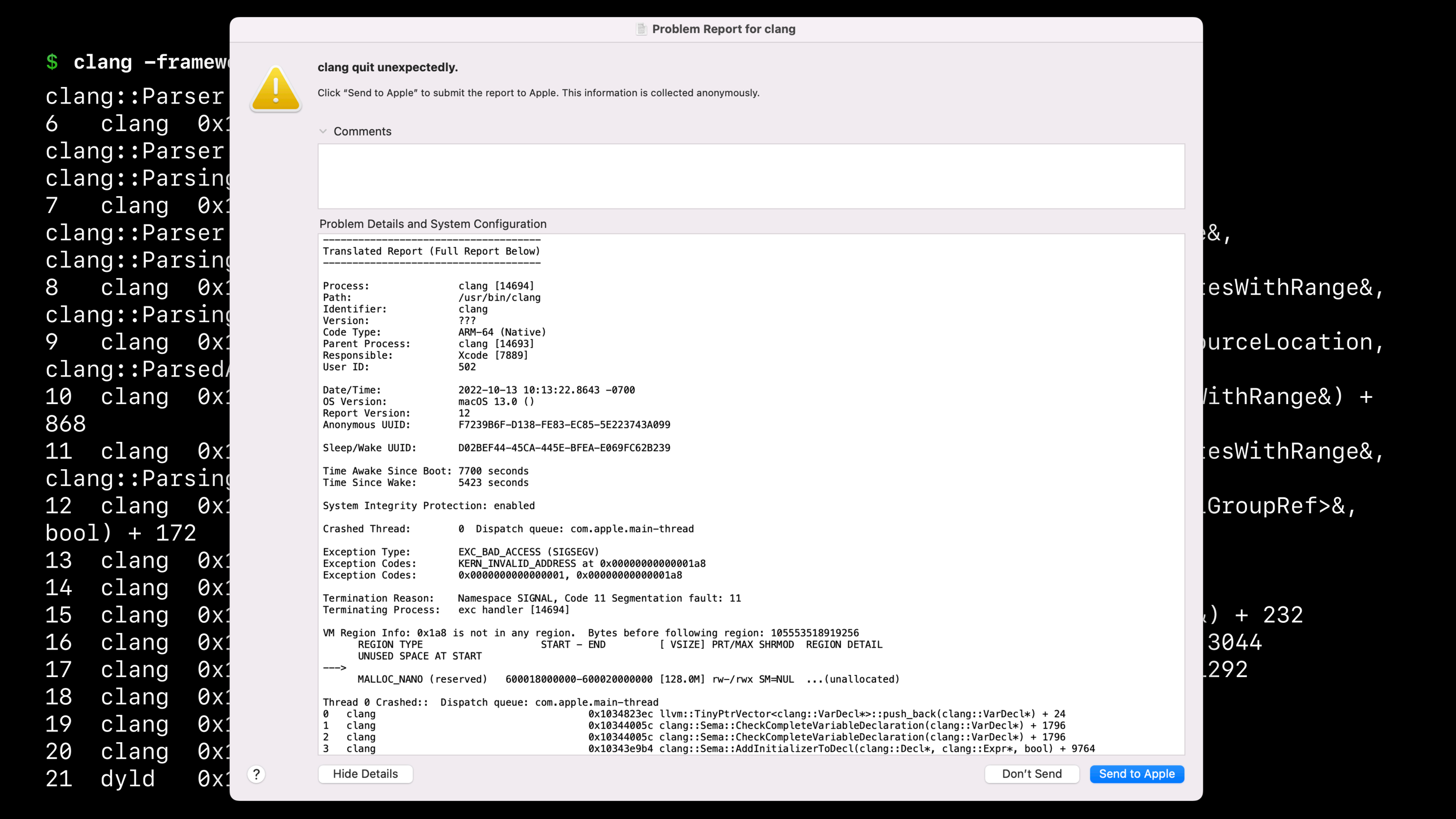
Interactive Crashlogs in LLDB

Ismail Bennani

LLVM Dev Meeting | Apple, Inc. | November 8-9th , 2022

$

```
$ clang -framework Foundation main.m
clang::Parser::LateParsedAttrList*) + 1600
6   clang  0x103231900
clang::Parser::ParseDeclOrFunctionDefInternal(clang::ParsedAttributesWithRange&,
clang::ParsingDeclSpec&, clang::AccessSpecifier) + 348
7   clang  0x1032313a4
clang::Parser::ParseDeclarationOrFunctionDefinition(clang::ParsedAttributesWithRange&,
clang::ParsingDeclSpec*, clang::AccessSpecifier) + 440
8   clang  0x103230368 clang::Parser::ParseExternalDeclaration(clang::ParsedAttributesWithRange&,
clang::ParsingDeclSpec*) + 140
9   clang  0x1031f09f0 clang::Parser::ParseObjCAtImplementationDeclaration(clang::SourceLocation,
clang::ParsedAttributes&) + 440
10  clang  0x1031efbe4 clang::Parser::ParseObjCAtDirectives(clang::ParsedAttributesWithRange&) +
868
11  clang  0x10323063c clang::Parser::ParseExternalDeclaration(clang::ParsedAttributesWithRange&,
clang::ParsingDeclSpec*) + 864
12  clang  0x10322f5ec clang::Parser::ParseTopLevelDecl(clang::OpaquePtr<clang::DeclGroupRef>&,
bool) + 172
13  clang  0x10318b0dc clang::ParseAST(clang::Sema&, bool, bool) + 256
14  clang  0x1021eb768 clang::FrontendAction::Execute() + 60
15  clang  0x102196690 clang::CompilerInstance::ExecuteAction(clang::FrontendAction&) + 232
16  clang  0x102245c30 clang::ExecuteCompilerInvocation(clang::CompilerInstance*) + 3044
17  clang  0x1007f7d68 cc1_main(llvm::ArrayRef<char const*>, char const*, void*) + 1292
18  clang  0x1007f57c4 ExecuteCC1Tool(llvm::SmallVectorImpl<char const*>&) + 708
19  clang  0x1007f5248 driver_main(int, char const**) + 13968
20  clang  0x1007f1430 main + 516
21  dyld   0x1b29efe50 start + 2544
```


$ clang -framew
clang::Parser
6 clang 0x1
clang::Parser
clang::Parsing
7 clang 0x1
clang::Parser
clang::Parsing
8 clang 0x1
clang::Parsing
9 clang 0x1
clang::Parsed
10 clang 0x1
868
11 clang 0x1
clang::Parsing
12 clang 0x1
bool) + 172
13 clang 0x1
14 clang 0x1
15 clang 0x1
16 clang 0x1
17 clang 0x1
18 clang 0x1
19 clang 0x1
20 clang 0x1
21 dyld 0x1
Problem Report for clang
clang quit unexpectedly.
Click "Send to Apple" to submit the report to Apple. This information is collected anonymously.
Comments
Problem Details and System Configuration

Translated Report (Full Report Below)

Process: clang [14694]
Path: /usr/bin/clang
Identifier: clang
Version: ???
Code Type: ARM-64 (Native)
Parent Process: clang [14693]
Responsible: Xcode [7889]
User ID: 502
Date/Time: 2022-10-13 10:13:22.8643 -0700
OS Version: macOS 13.0 ()
Report Version: 12
Anonymous UUID: F7239B6F-D138-FE83-EC85-5E223743A099
Sleep/Wake UUID: D02BEF44-45CA-445E-BFEA-E069FC62B239
Time Awake Since Boot: 7700 seconds
Time Since Wake: 5423 seconds
System Integrity Protection: enabled
Crashed Thread: 0 Dispatch queue: com.apple.main-thread
Exception Type: EXC_BAD_ACCESS (SIGSEGV)
Exception Codes: KERN_INVALID_ADDRESS at 0x00000000000001a8
Exception Codes: 0x0000000000000001, 0x00000000000001a8
Termination Reason: Namespace SIGNAL, Code 11 Segmentation fault: 11
Terminating Process: exc handler [14694]
VM Region Info: 0x1a8 is not in any region. Bytes before following region: 105553518919256
REGION TYPE START - END [VSIZE] PRT/MAX SHRMOD REGION DETAIL
UNUSED SPACE AT START
--->
MALLOC_NANO (reserved) 600018000000-600020000000 [128.0M] rw-/rwx SM=NUL ...(unallocated)
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0 clang 0x1034823ec llvm::TinyPtrVector<clang::VarDecl*>::push_back(clang::VarDecl*) + 24
1 clang 0x10344005c clang::Sema::CheckCompleteVariableDeclaration(clang::VarDecl*) + 1796
2 clang 0x10344005c clang::Sema::CheckCompleteVariableDeclaration(clang::VarDecl*) + 1796
3 clang 0x10343e9b4 clang::Sema::AddInitializerToDecl(clang::Decl*, clang::Expr*, bool) + 9764
Hide Details
Don't Send
Send to Apple

Problem Report for clang

clang quit unexpectedly.

Click “Send to Apple” to submit the report to Apple. This information is collected anonymously.

Comments

Problem Details and System Configuration

```
----------------------------------------
Translated Report (Full Report Below)
----------------------------------------

Process:               clang [14694]
Path:                  /usr/bin/clang
Identifier:            clang
Version:               ???
Code Type:             ARM-64 (Native)
Parent Process:        clang [14693]
Responsible:           Xcode [7889]
User ID:               502

Date/Time:             2022-10-13 10:13:22.8643 -0700
OS Version:            macOS 13.0 ()
Report Version:        12
Anonymous UUID:        F7239B6F-D138-FE83-EC85-5E223743A099

Sleep/Wake UUID:       D02BEF44-45CA-445E-BFEA-E069FC62B239

Time Awake Since Boot: 7700 seconds
Time Since Wake:       5423 seconds

System Integrity Protection: enabled

Crashed Thread:        0  Dispatch queue: com.apple.main-thread

Exception Type:        EXC_BAD_ACCESS (SIGSEGV)
Exception Codes:       KERN_INVALID_ADDRESS at 0x00000000000001a8
Exception Codes:       0x0000000000000001, 0x00000000000001a8

Termination Reason:    Namespace SIGNAL, Code 11 Segmentation fault: 11
Terminating Process:   exc handler [14694]

VM Region Info: 0x1a8 is not in any region.  Bytes before following region: 105553518919256
      REGION TYPE                    START - END         [ VSIZE] PRT/MAX SHRMOD  REGION DETAIL
      UNUSED SPACE AT START
--->
      MALLOC_NANO (reserved)   600018000000-600020000000 [128.0M] rw-/rwx SM=NUL  ...(unallocated)

Thread 0 Crashed::  Dispatch queue: com.apple.main-thread
0   clang                               0x1034823ec llvm::TinyPtrVector<clang::VarDecl*>::push_back(clang::VarDecl*) + 24
1   clang                               0x10344005c clang::Sema::CheckCompleteVariableDeclaration(clang::VarDecl*) + 1796
2   clang                               0x10344005c clang::Sema::CheckCompleteVariableDeclaration(clang::VarDecl*) + 1796
3   clang                               0x10343e9b4 clang::Sema::AddInitializerToDecl(clang::Decl*, clang::Expr*, bool) + 9764
```

?

Hide Details

Don’t Send

Send to Apple

```
$ cat clang.crash
```

```
$ cat clang.crash

{
  "bug_type": "309",
  "captureTime": "2022-10-13 10:13:22.8643 -0700",
  "coalitionName": "com.apple.dt.Xcode",
  "cpuType": “ARM-64",

  "exception": {
    "codes": "0x0000000000000001, 0x00000000000001a8",
    "rawCodes": [
      1,
      424
    ],
    "signal": "SIGSEGV",
    "subtype": "KERN_INVALID_ADDRESS at 0x00000000000001a8",
    "type": "EXC_BAD_ACCESS"
  },

  "faultingThread": 0,
  "incident": "AFE4747E-DAD6-4426-AAC6-088771938CA1",
  "legacyInfo": {
    "threadTriggered": {
      "queue": "com.apple.main-thread"
    }
  },
  "modelCode": "MacBookPro18,1",
```

```
$ cat clang.crash

{
  "bug_type": "309",
  "captureTime": "2022-10-13 10:13:22.8643 -0700",
  "coalitionName": "com.apple.dt.Xcode",
  "cpuType": “ARM-64",

  "exception": {
    "codes": "0x0000000000000001, 0x00000000000001a8",
    "rawCodes": [
      1,
      424
    ],
    "signal": "SIGSEGV",
    "subtype": "KERN_INVALID_ADDRESS at 0x00000000000001a8",
    "type": "EXC_BAD_ACCESS"
  },

  "faultingThread": 0,
  "incident": "AFE4747E-DAD6-4426-AAC6-088771938CA1",
  "legacyInfo": {
    "threadTriggered": {
      "queue": "com.apple.main-thread"
    }
  },
  "modelCode": "MacBookPro18,1",
```

```
$ cat clang.crash
```

```
"threads": [
  {
    "frames": [
      {
        "imageIndex": 0,
        "imageOffset": 46752748,
        "symbol": "llvm::TinyPtrVector<clang::VarDecl*>::push_back(clang::VarDecl*)",
        "symbolLocation": 24
      },
      {
        "imageIndex": 0,
        "imageOffset": 46481500,
        "symbol": "clang::Sema::CheckCompleteVariableDeclaration(clang::VarDecl*)",
        "symbolLocation": 1796
      },

      . . .
    ],

    "id": 261291,
    "queue": “com.apple.main-thread",

    "threadState": {
      "flavor": "ARM_THREAD_STATE64",
```

$ cat clang.crash

```
    "threadState": {
      "flavor": "ARM_THREAD_STATE64",
      "cpsr": { "value": 1610616832 },
      "esr": { "value": 2449473542 },
      "far": { "value": 424 },
      "fp": { "value": 6163527520 },
      "lr": { "value": 4349755484 },
      "pc": { "value": 4350026732 },
      "sp": { "value": 6163527488 },
      "x": [
        { "value": 424 },
        { "value": 5510078088 },
        { "value": 1 },
        { "value": 4999690240 },
        { "value": 6163528328 },
        { "value": 0 }
        ]
     },
    "triggered": true
  }
],

"translated": false,
"uptime": 7700,
```

```
$ cat clang.crash
  "uptime": 7700,
  "usedImages": [
    {
      "arch": "arm64",
      "base": 4303273984,
      "name": "clang",
      "path": "/usr/bin/clang",
      "size": 74104832,
      "source": "P",
      "uuid": "e109a61a-7ac7-37ba-82c3-56a72eb5362e"
    },
    {
      "arch": "arm64e",
      "base": 7291707392,
      "name": "dyld",
      "path": "/usr/lib/dyld",
      "size": 566448,
      "source": "P",
      "uuid": "1f56ab4e-f398-3a9c-bfba-fbbd17808963"
    },
  ],
  "userID": 502,
  "version": 2,
}
```

```
(lldb) command script import lldb.macosx.crashlog

(lldb) crashlog clang.crash


Thread[0] EXC_BAD_ACCESS (SIGSEGV) (0x0000000000000001, 0x00000000000001a8)
[  0] 0x00000001034823ec clang`::push_back() [inlined] getPointer at PointerIntPair.h:59:58
       55    }
       56
       57    explicit PointerIntPair(PointerTy PtrVal) { initWithPointer(PtrVal); }
       58
    -> 59    PointerTy getPointer() const { return Info::getPointer(Value); }
       60
       61    IntType getInt() const { return (IntType)Info::getInt(Value); }
       62
       63    void setPointer(PointerTy PtrVal) LLVM_LVALUE_FUNCTION {


     0x00000001034823dc:      stp x29, x30, [sp, #0x20]
     0x00000001034823e0:      add x29, sp, #0x20
     0x00000001034823e4:      mov x19, x1
     0x00000001034823e8:      mov x20, x0
 ->  0x00000001034823ec:      ldr x8, [x0]
     0x00000001034823f0:      cmp x8, #0x7
     0x00000001034823f4:     b.hi 0x102c96410          ; <+60> at TinyPtrVector.h:253:9
     0x00000001034823f8:      and x8, x19, #0xfffffffffffffffb
     0x00000001034823fc:      str x8, [x20]

[  0] 0x00000001034823ec clang`::push_back() [inlined] isNull at PointerUnion.h:143:43
       139   using Base::Base;
       140
       141   /// Test if the pointer held in the union is null, regardless of
       142   /// which type it is.
    -> 143   bool isNull() const { return !this->Val.getPointer(); }
       144
       145   explicit operator bool() const { return !isNull(); }
       146
       147   /// Test if the Union currently holds the type matching T.

[  0] 0x00000001034823ec clang`::push_back() + 24 at TinyPtrVector.h:246:13
       242   }
       243
       244   void push_back(EltTy NewVal) {
       245     // If we have nothing, add something.
    -> 246     if (Val.isNull()) {
       247       Val = NewVal;
       248       assert(!Val.isNull() && "Can't add a null value");
```

```
(lldb) command script import lldb.macosx.crashlog
(lldb) crashlog --interactive clang.crash
(lldb)
```

```
(lldb) command script import lldb.macosx.crashlog

(lldb) crashlog --interactive clang.crash

(lldb) process status
Process 14694 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x1a8)
    frame #0: 0x00000001034823ec clang`::push_back() [inlined] getPointer at PointerIntPair.h:59:58 [opt] [artificial]
   56
   57    explicit PointerIntPair(PointerTy PtrVal) { initWithPointer(PtrVal); }
   58
-> 59    PointerTy getPointer() const { return Info::getPointer(Value); }
   60
   61    IntType getInt() const { return (IntType)Info::getInt(Value); }
   62
```

```
(lldb) command script import lldb.macosx.crashlog

(lldb) crashlog --interactive clang.crash

(lldb) process status

(lldb) thread backtrace
* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x1a8)
  * frame #0: 0x00000001034823ec clang`::push_back() [inlined] getPointer at PointerIntPair.h:59:58 [opt] [artificial]
    frame #1: 0x000000010344005b clang`::CheckCompleteVariableDeclaration() [inlined] addByrefBlockVar at ScopeInfo.h:477:20
[opt] [artificial]
    frame #2: 0x000000010344005b clang`::CheckCompleteVariableDeclaration() [inlined] addByrefBlockVar at ScopeInfo.h:477:20
[opt] [artificial]
    frame #3: 0x000000010343e9b3 clang`::AddInitializerToDecl() at SemaDecl.cpp:13259:3 [opt] [artificial]
    frame #4: 0x00000001031a61f3 clang`::ParseDeclarationAfterDeclaratorAndAttributes() at ParseDecl.cpp:0 [opt] [artificial]
    frame #5: 0x00000001031a46bb clang`::ParseDeclGroup() at ParseDecl.cpp:2180:21 [opt] [artificial]
    frame #6: 0x00000001032318ff clang`::ParseDeclOrFunctionDefInternal() at Parser.cpp:1148:10 [opt] [artificial]
    frame #7: 0x00000001032313a3 clang`::ParseDeclarationOrFunctionDefinition() at Parser.cpp:1165:12 [opt] [artificial]
    frame #8: 0x0000000103230367 clang`::ParseExternalDeclaration() at Parser.cpp:984:12 [opt] [artificial]
    frame #9: 0x00000001031f09ef clang`::ParseObjCAtImplementationDeclaration() at ParseObjc.cpp:2229:32 [opt] [artificial]
    frame #10: 0x00000001031efbe3 clang`::ParseObjCAtDirectives() at ParseObjc.cpp:68:12 [opt] [artificial]
    frame #11: 0x000000010323063b clang`::ParseExternalDeclaration() at Parser.cpp:876:12 [opt] [artificial]
    frame #12: 0x000000010322f5eb clang`::ParseTopLevelDecl() at Parser.cpp:728:12 [opt] [artificial]
    frame #13: 0x000000010318b0db clang`::ParseAST() at ParseAST.cpp:158:20 [opt] [artificial]
    frame #14: 0x00000001021eb767 clang`::Execute() at FrontendAction.cpp:971:8 [opt] [artificial]
    frame #15: 0x000000010219668f clang`::ExecuteAction() at CompilerInstance.cpp:1138:33 [opt] [artificial]
    frame #16: 0x0000000102245c2f clang`::ExecuteCompilerInvocation() at ExecuteCompilerInvocation.cpp:288:25 [opt] [artificial]
    frame #17: 0x00000001007f7d67 clang`::cc1_main() at cc1_main.cpp:246:15 [opt] [artificial]
    frame #18: 0x00000001007f57c3 clang`::ExecuteCC1Tool() at driver.cpp:319:12 [opt] [artificial]
    frame #19: 0x00000001007f5247 clang`::driver_main() at driver.cpp:456:12 [opt] [artificial]
    frame #20: 0x00000001007f142f clang`main at driver.cpp:373:16 [opt] [artificial]
```

```
(lldb) command script import lldb.macosx.crashlog
(lldb) crashlog --interactive clang.crash
(lldb) process status
(lldb) thread backtrace
```

```
(lldb) command script import lldb.macosx.crashlog

(lldb) crashlog --interactive clang.crash

(lldb) process status

(lldb) thread backtrace

(lldb) frame select 13
frame #13: 0x000000010318b0db clang`::ParseAST() at ParseAST.cpp:158:20 [opt] [artificial]
   155      P.Initialize();
   156      Parser::DeclGroupPtrTy ADecl;
   157      for (bool AtEOF = P.ParseFirstTopLevelDecl(ADecl); !AtEOF;
-> 158           AtEOF = P.ParseTopLevelDecl(ADecl)) {
   159        // If we got a null return and something *was* parsed, ignore it.  This
   160        // is due to a top-level semicolon, an action override, or a parse error
   161        // skipping something.
```

```
(lldb) command script import lldb.macosx.crashlog

(lldb) crashlog --interactive clang.crash

(lldb) process status

(lldb) thread backtrace

(lldb) frame select 13
frame #13: 0x000000010318b0db clang`::ParseAST() at ParseAST.cpp:158:20 [opt] [artificial]
   155      P.Initialize();
   156      Parser::DeclGroupPtrTy ADecl;
   157      for (bool AtEOF = P.ParseFirstTopLevelDecl(ADecl); !AtEOF;
-> 158           AtEOF = P.ParseTopLevelDecl(ADecl)) {
   159        // If we got a null return and something *was* parsed, ignore it.  This
   160        // is due to a top-level semicolon, an action override, or a parse error
   161        // skipping something.
(lldb) source list -c 10
   162        if (ADecl && !Consumer->HandleTopLevelDecl(ADecl.get()))
   163          return;
   164      }
   165    }
   166
   167    // Process any TopLevelDecls generated by #pragma weak.
   168    for (Decl *D : S.WeakTopLevelDecls())
   169      Consumer->HandleTopLevelDecl(DeclGroupRef(D));
   170
   171    Consumer->HandleTranslationUnit(S.getASTContext());
```

```
(lldb) command script import lldb.macosx.crashlog
(lldb) crashlog --interactive clang.crash
(lldb) process status
(lldb) thread backtrace
(lldb) frame select 13
(lldb) source list -c 10
```

```
(lldb) command script import lldb.macosx.crashlog
(lldb) crashlog --interactive clang.crash
(lldb) process status
(lldb) thread backtrace
(lldb) frame select 13
(lldb) source list -c 10
(lldb) disassemble -a 0x0000000103440050b -c 20
```

```
(lldb) command script import lldb.macosx.crashlog
(lldb) crashlog --interactive clang.crash
(lldb) process status
(lldb) thread backtrace
(lldb) frame select 13
(lldb) source list -c 10
(lldb) disassemble -a 0x0000000103440058b -c 20
clang`::CheckCompleteVariableDeclaration():
    0x10343f958 <+0>:  stp    x28, x27, [sp, #-0x60]!
    0x10343f95c <+4>:  stp    x26, x25, [sp, #0x10]
    0x10343f960 <+8>:  stp    x24, x23, [sp, #0x20]
    0x10343f964 <+12>: stp    x22, x21, [sp, #0x30]
    0x10343f968 <+16>: stp    x20, x19, [sp, #0x40]
    0x10343f96c <+20>: stp    x29, x30, [sp, #0x50]
    0x10343f970 <+24>: add    x29, sp, #0x50
    0x10343f974 <+28>: sub    sp, sp, #0x400
    0x10343f978 <+32>: stur   x1, [x29, #-0x60]
    0x10343f97c <+36>: ldrb   w8, [x1, #0x1c]
    0x10343f980 <+40>: tbnz   w8, #0x7, 0x102c53db0     ; <+1112> at SemaDecl.cpp:13984:1
    0x10343f984 <+44>: mov    x20, x1
    0x10343f988 <+48>: mov    x19, x0
    0x10343f98c <+52>: bl     0x102c14d5c               ; ::MaybeAddCUDAConstantAttr() at SemaCUDA.cpp:712
    0x10343f990 <+56>: ldr    x8, [x19, #0x40]
    0x10343f994 <+60>: ldr    w9, [x8, #0x30]
    0x10343f998 <+64>: tbnz   w9, #0x1d, 0x102c5446c    ; <+2836> [inlined] getInt at PointerIntPair.h:61:57
    0x10343f99c <+68>: ldrb   w8, [x8, #0x1]
    0x10343f9a0 <+72>: tbnz   w8, #0x6, 0x102c53ef4     ; <+1436> at SemaDecl.cpp:13698:12
    0x10343f9a4 <+76>: mov    x0, x20
```

```
(lldb) command script import lldb.macosx.crashlog

(lldb) crashlog --interactive clang.crash

(lldb) process status

(lldb) thread backtrace

(lldb) frame select 13

(lldb) source list -c 10

(lldb) disassemble -a 0x0000000103440050b -c 20

(lldb) register read

General Purpose Registers:
        x0 = 0x00000000000001a8
        x1 = 0x00000001486d1e88
        x2 = 0x0000000000000001
        x3 = 0x000000012a013800
        x4 = 0x0000000000000000
        x5 = 0x0000000000000001
        x6 = 0x0000000000000004
        x7 = 0x000000016f5ffa88
         . . .
       x17 = 0x00000001031e0a4c  clang`::callback_fn<(lambda at ~/clang/lib/Parse/ParseExpr.cpp:2110:56)>() at STLExtras.h:176
        sp = 0x000000016f5ff740
        pc = 0x00000001034823ec  clang`::push_back() + 24 [inlined] getPointer at PointerIntPair.h:59:58
                                 clang`::push_back() + 24 [inlined] isNull at PointerUnion.h:143:43
                                 clang`::push_back() + 24 at TinyPtrVector.h:246:13
      cpsr = 0x60001000
```

The Big Picture

(lldb) ⟷ ?? ⟷ 

Crash Report

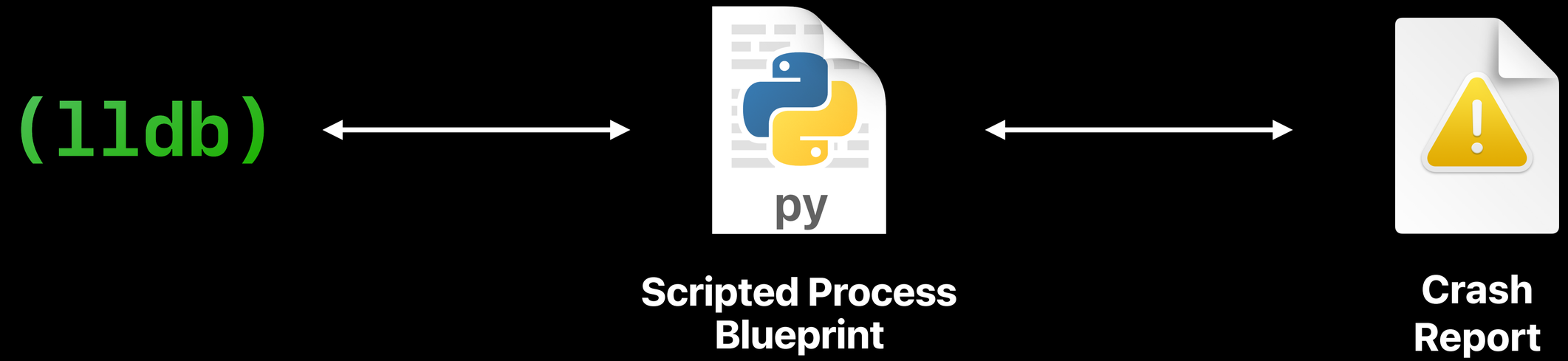
Scripted Process Blueprint

```
$ cat $LLDB/example/python/scripted_process/crashlog_scripted_process.py
```

```python
class CrashLogScriptedProcess(ScriptedProcess):
    def parse_crashlog(self):
        crashlog_parser = CrashLogParser(self.dbg, self.crashlog_path, False)
        crash_log = crashlog_parser.parse()

        self.pid = crash_log.process_id
        self.addr_mask = crash_log.addr_mask
        self.crashed_thread_idx = crash_log.crashed_thread_idx
        self.loaded_images = []
        self.exception = crash_log.exception

        def load_images(self, images):
            if images:
                for image in images:
                    if image not in self.loaded_images:
                        if image.uuid == uuid.UUID(int=0):
                            continue
                        err = image.add_module(self.target)
                        if err:
                            print(err)
                        else:
                            self.loaded_images.append(image)

        for thread in crash_log.threads:
            if self.load_all_images:
                load_images(self, crash_log.images)
            elif thread.did_crash():
                for ident in thread.idents:
                    load_images(self, crash_log.find_images_with_identifier(ident))

            self.threads[thread.index] = CrashLogScriptedThread(self, None, thread)
```

Scripted Process Blueprint

The Big Picture



The Big Picture

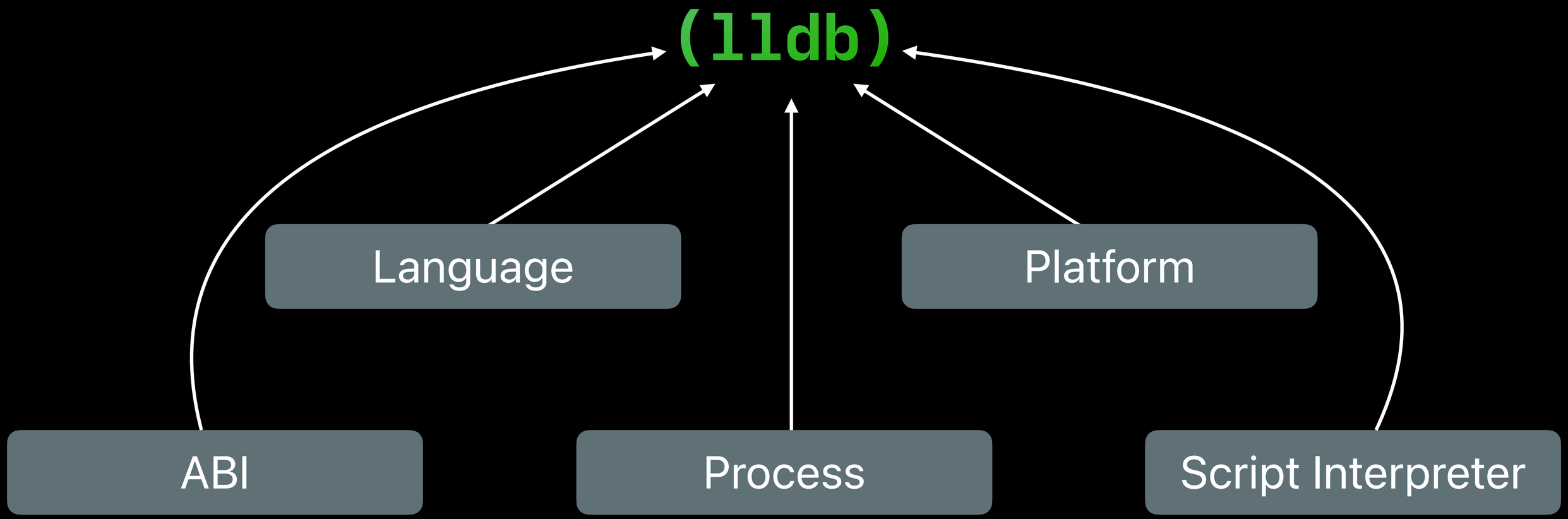
Scripted Process Plugin

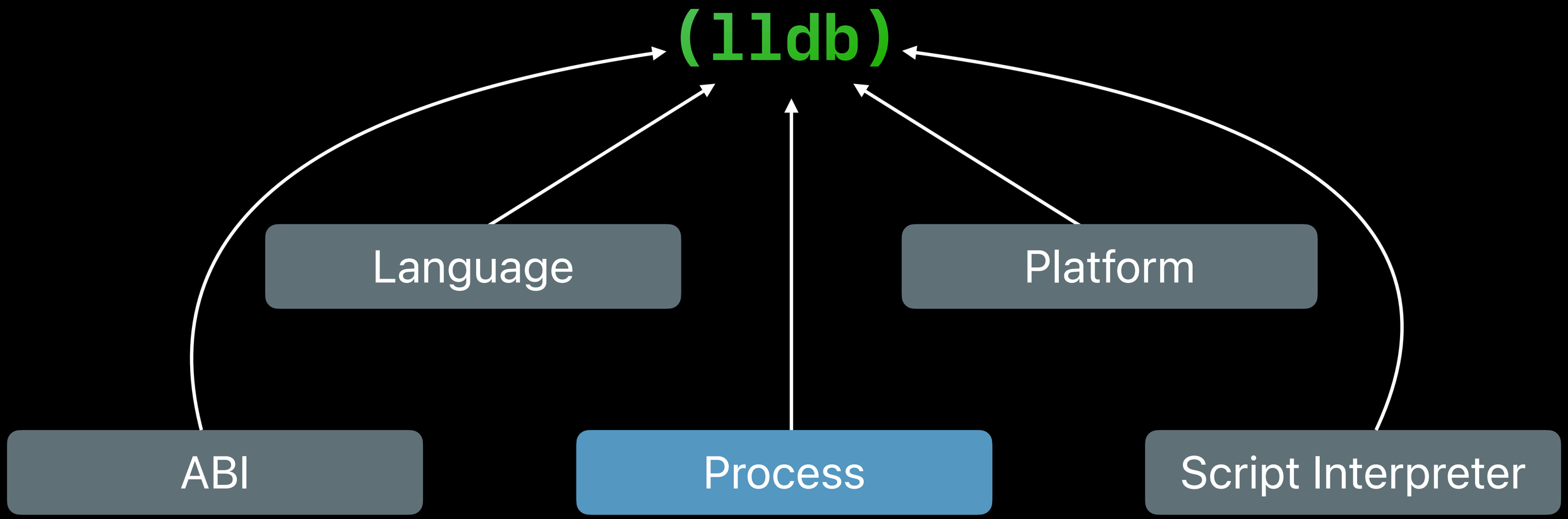
Scripted Process Blueprint

Crash Report

LLDB Process Plugin Model

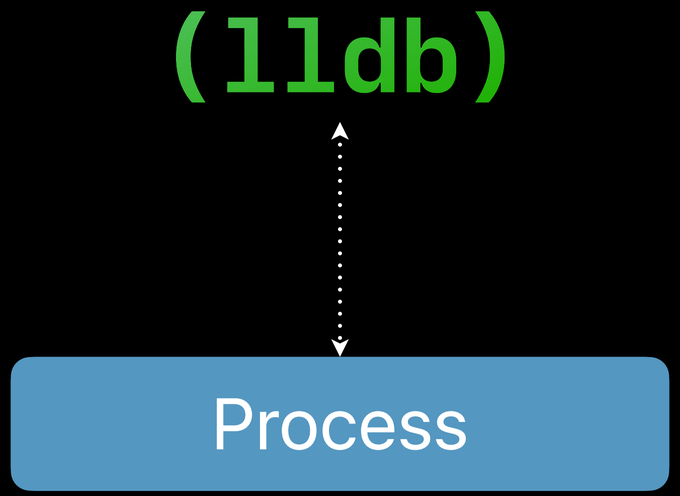
(lldb)

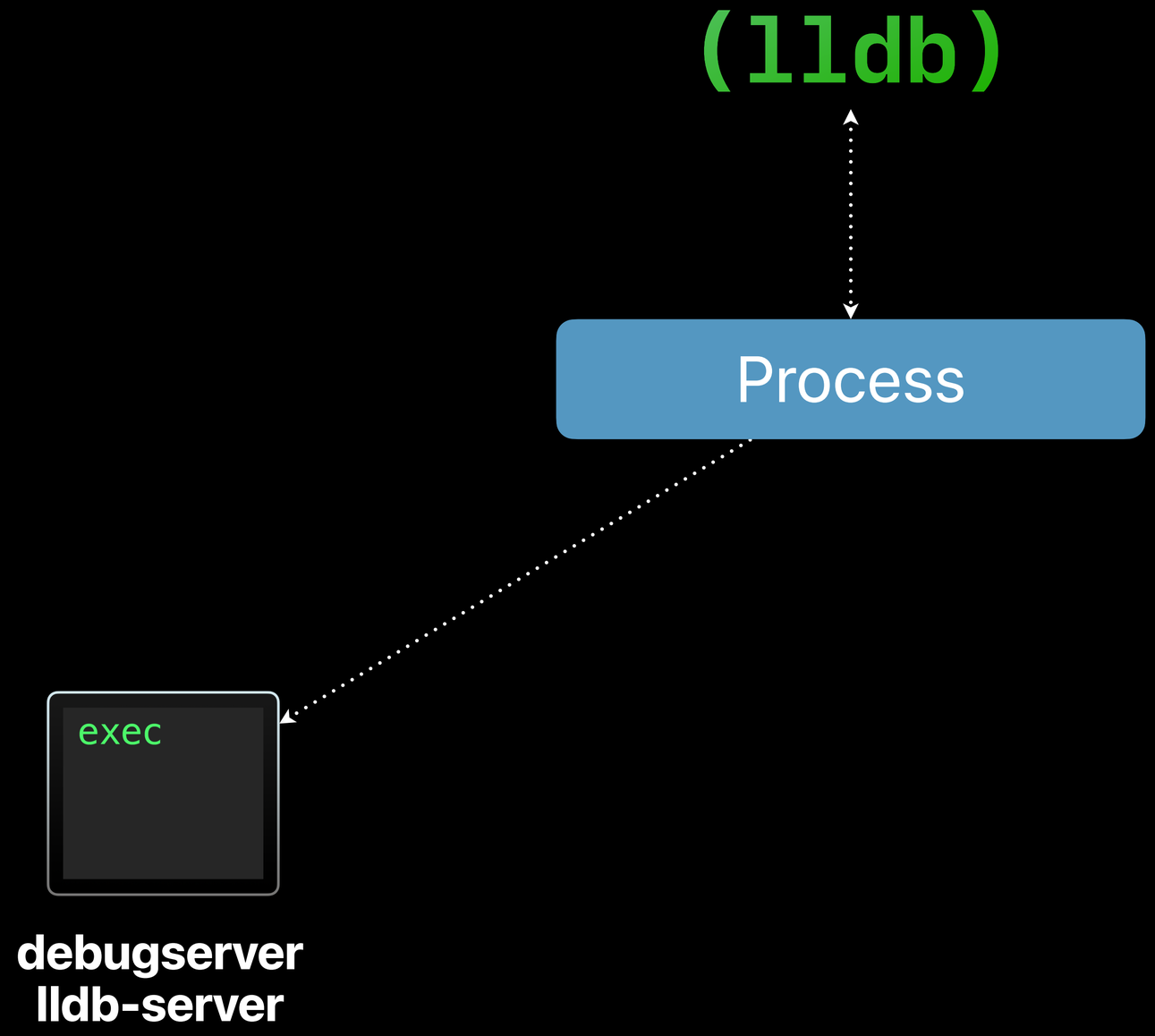

(lldb)
Language
Platform
ABI
Process
Script Interpreter

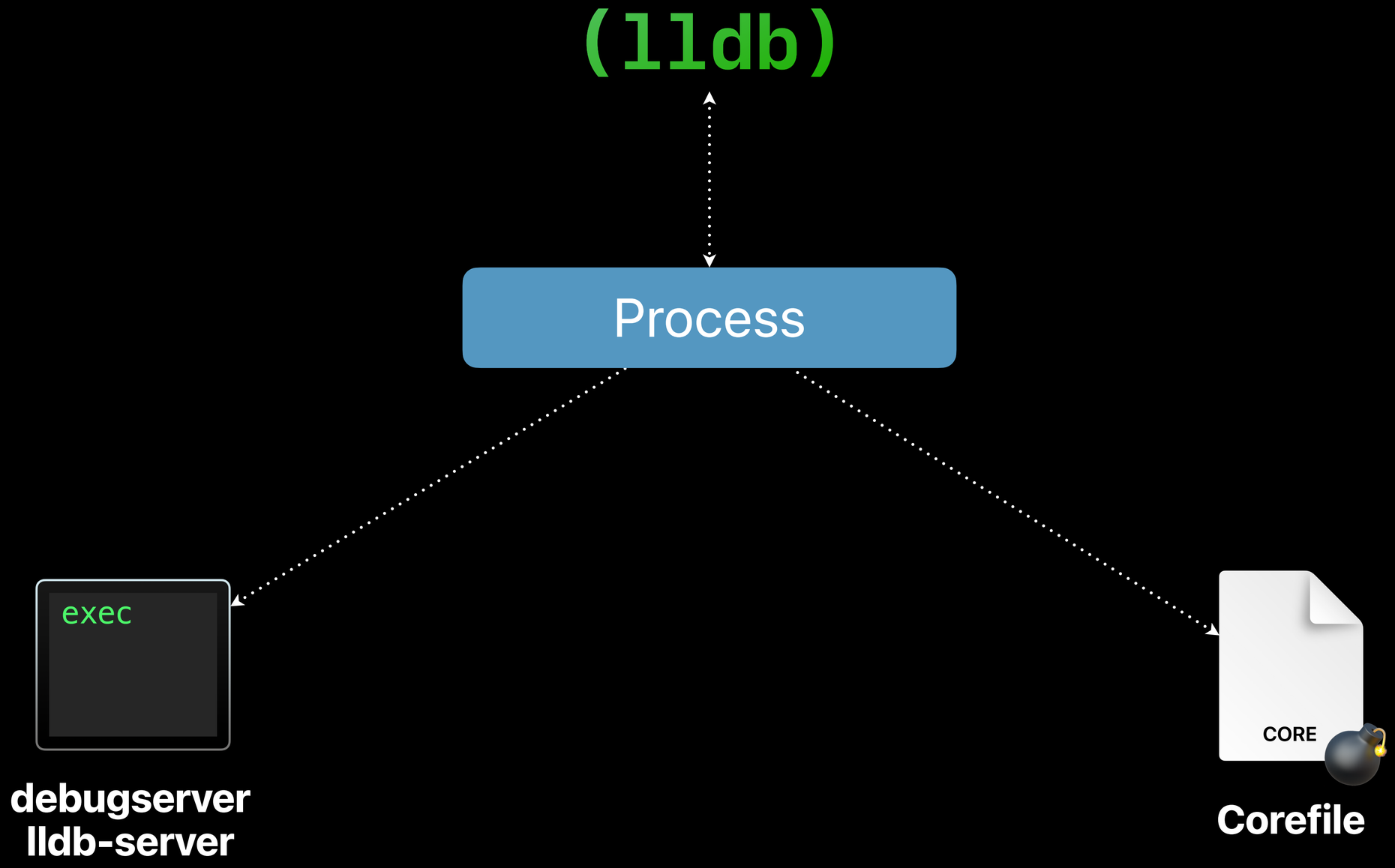

(lldb)
Language
Platform
ABI
Process
Script Interpreter

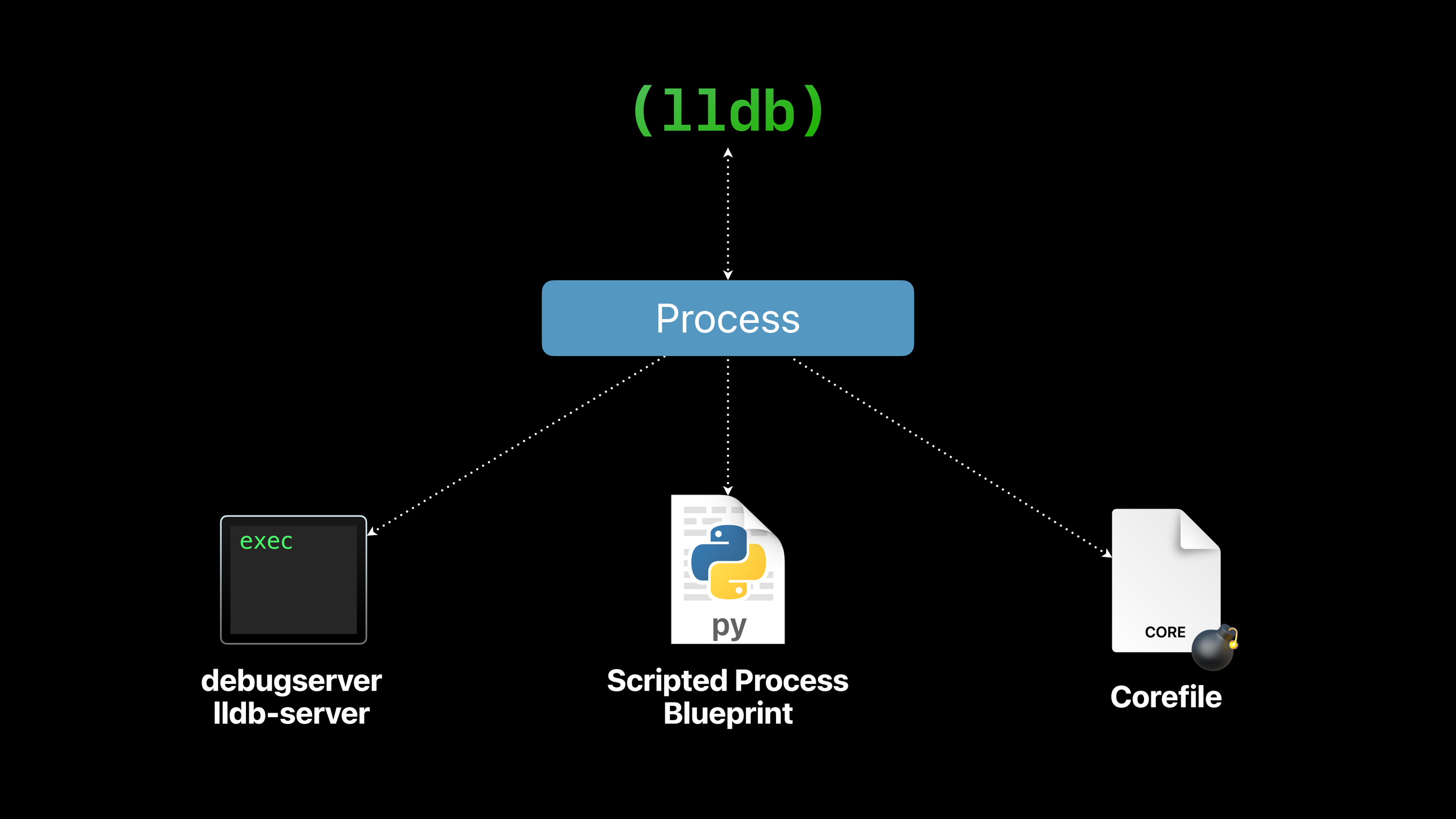
(lldb)

Process

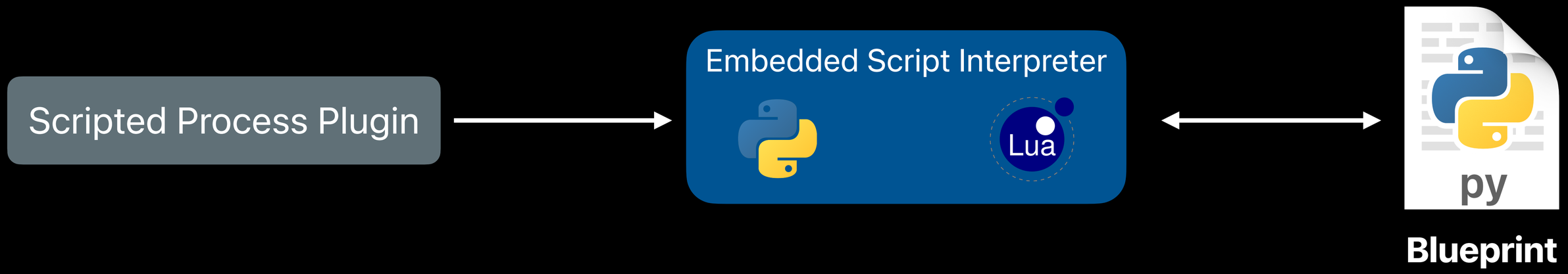

(lldb)
Process


(lldb)
Process
exec
debugserver
lldb-server

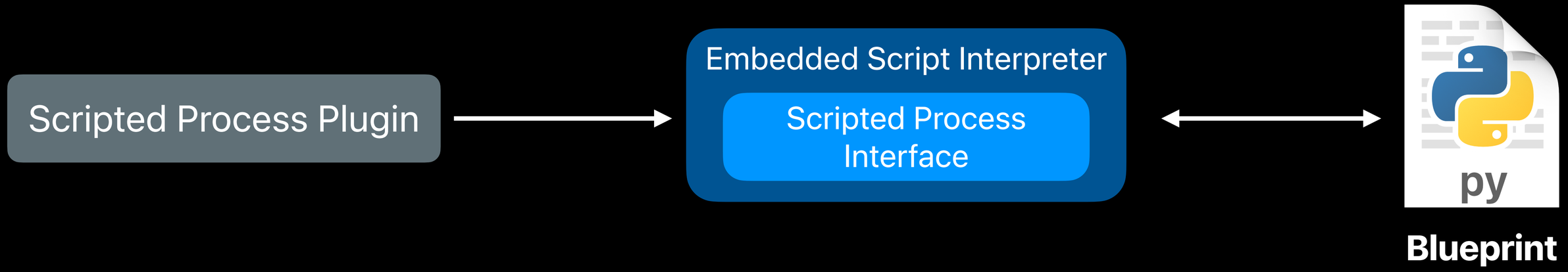

(lldb)
Process
exec
debugserver
lldb-server
CORE
Corefile


(lldb)
Process
exec
debugserver
lldb-server
py
Scripted Process
Blueprint
CORE
Corefile

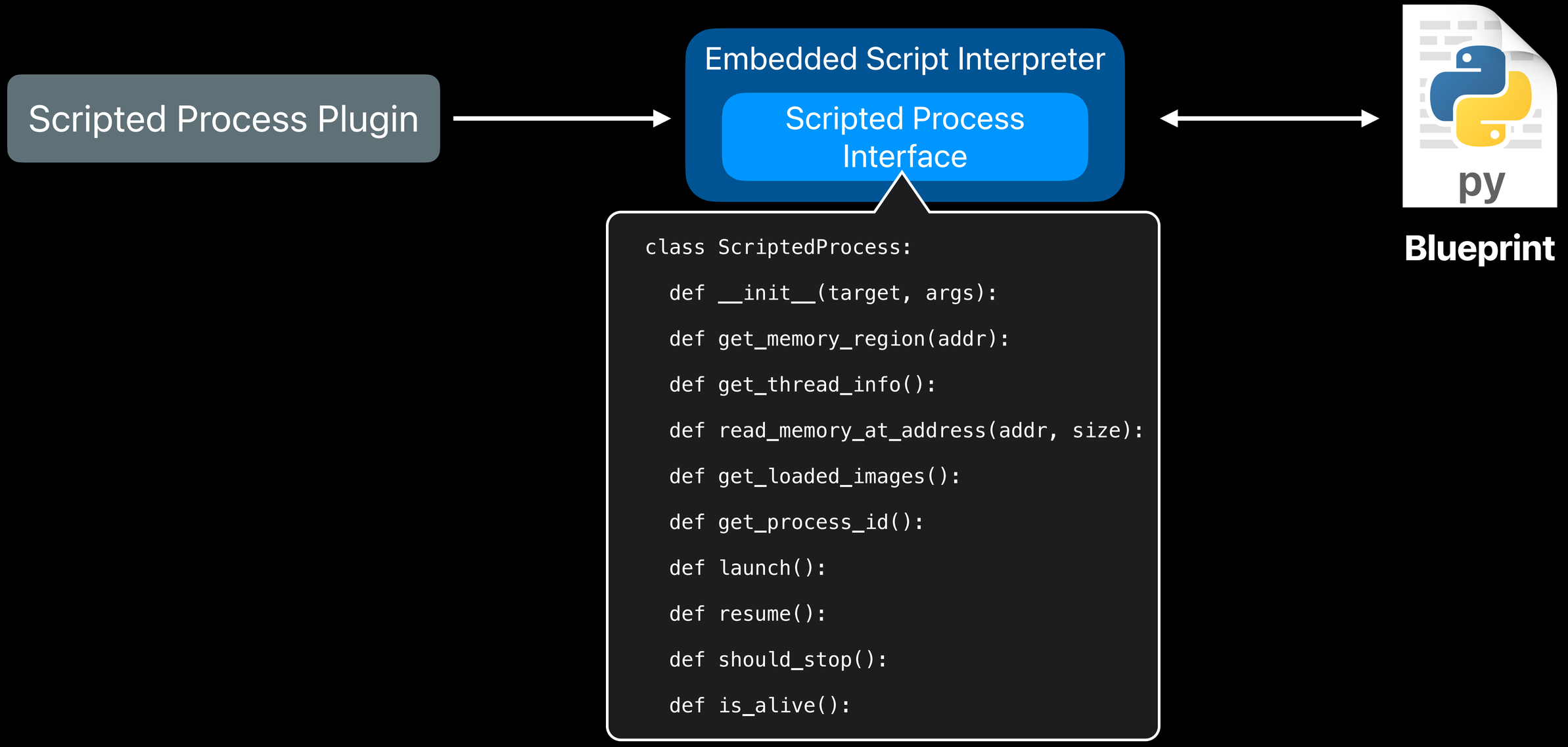
```
(lldb) process launch --script-class crashlog_scripted_process.CrashLogScriptedProcess
```

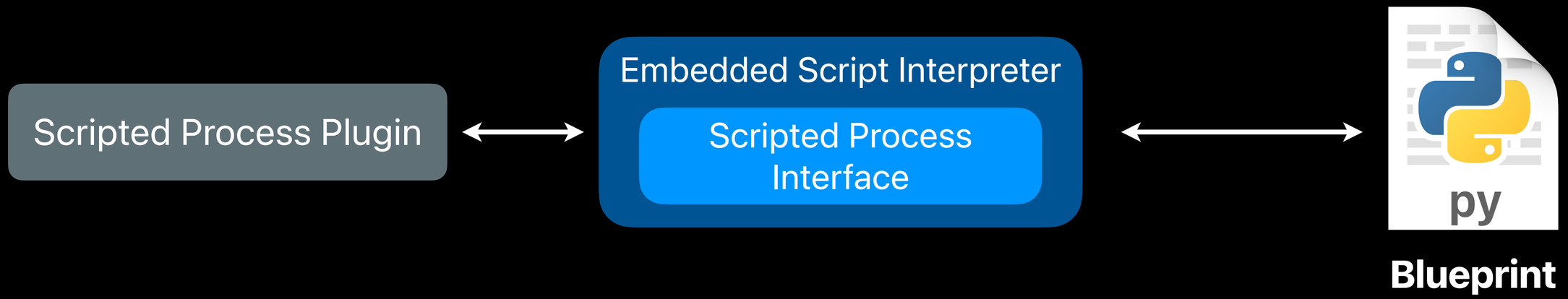


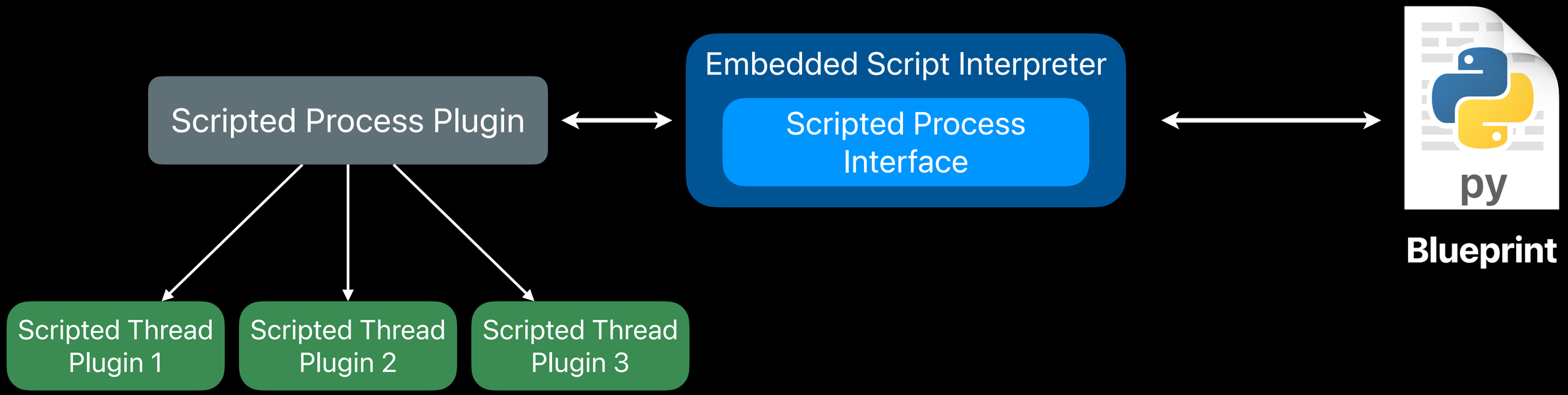
```
(lldb) process launch --script-class crashlog_scripted_process.CrashLogScriptedProcess
```

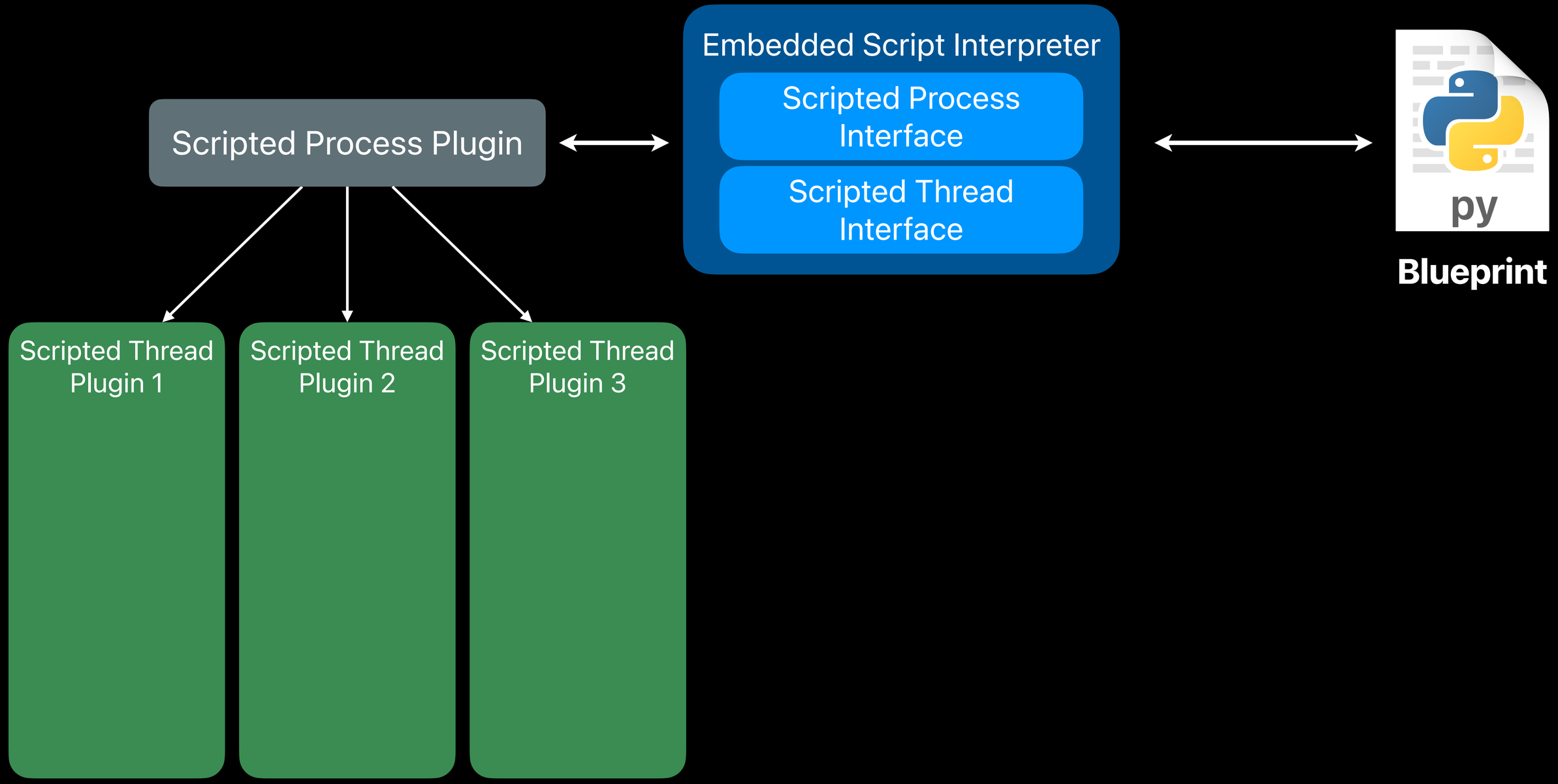


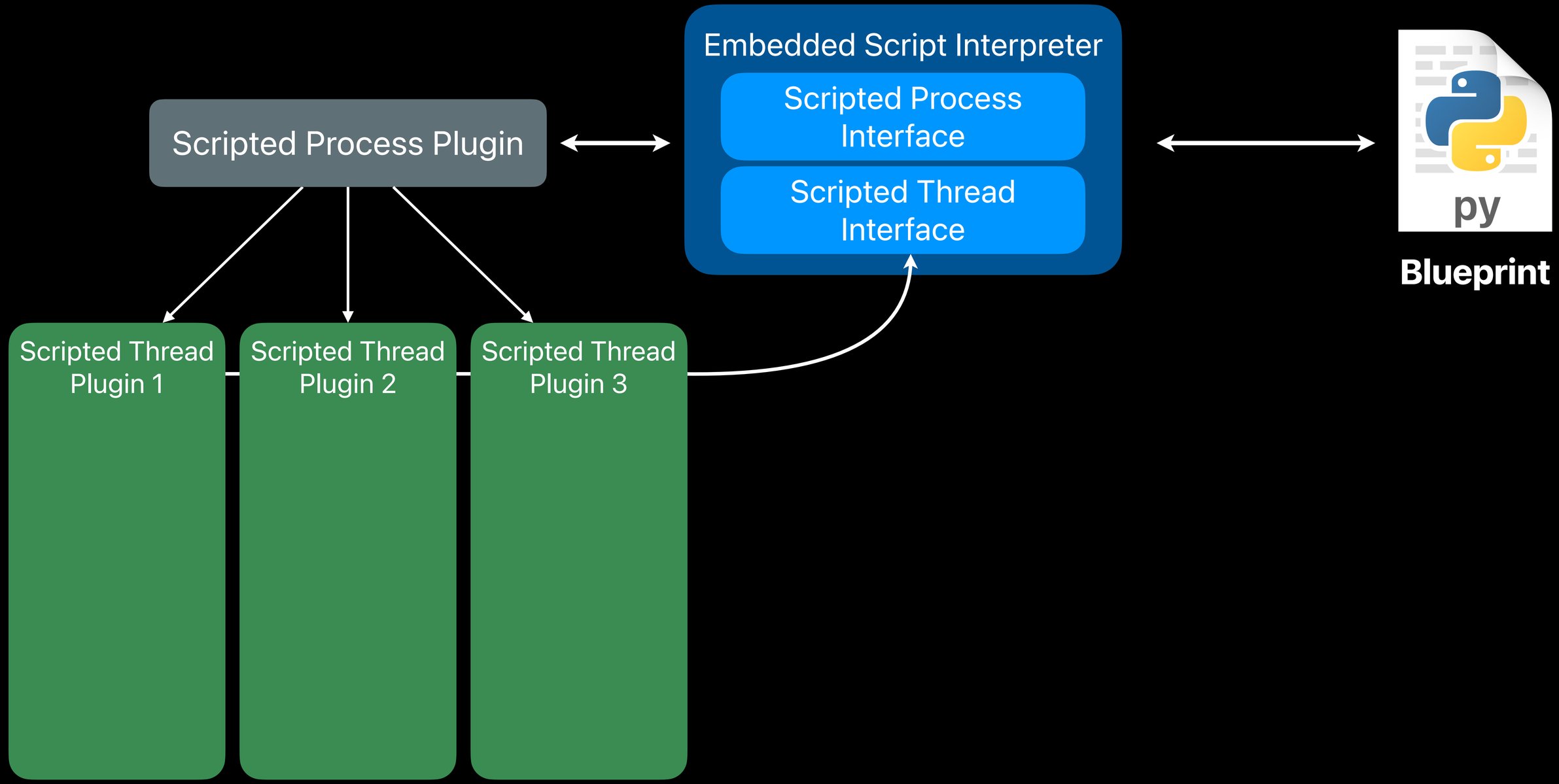
```
(lldb) process launch --script-class crashlog_scripted_process.CrashLogScriptedProcess
```

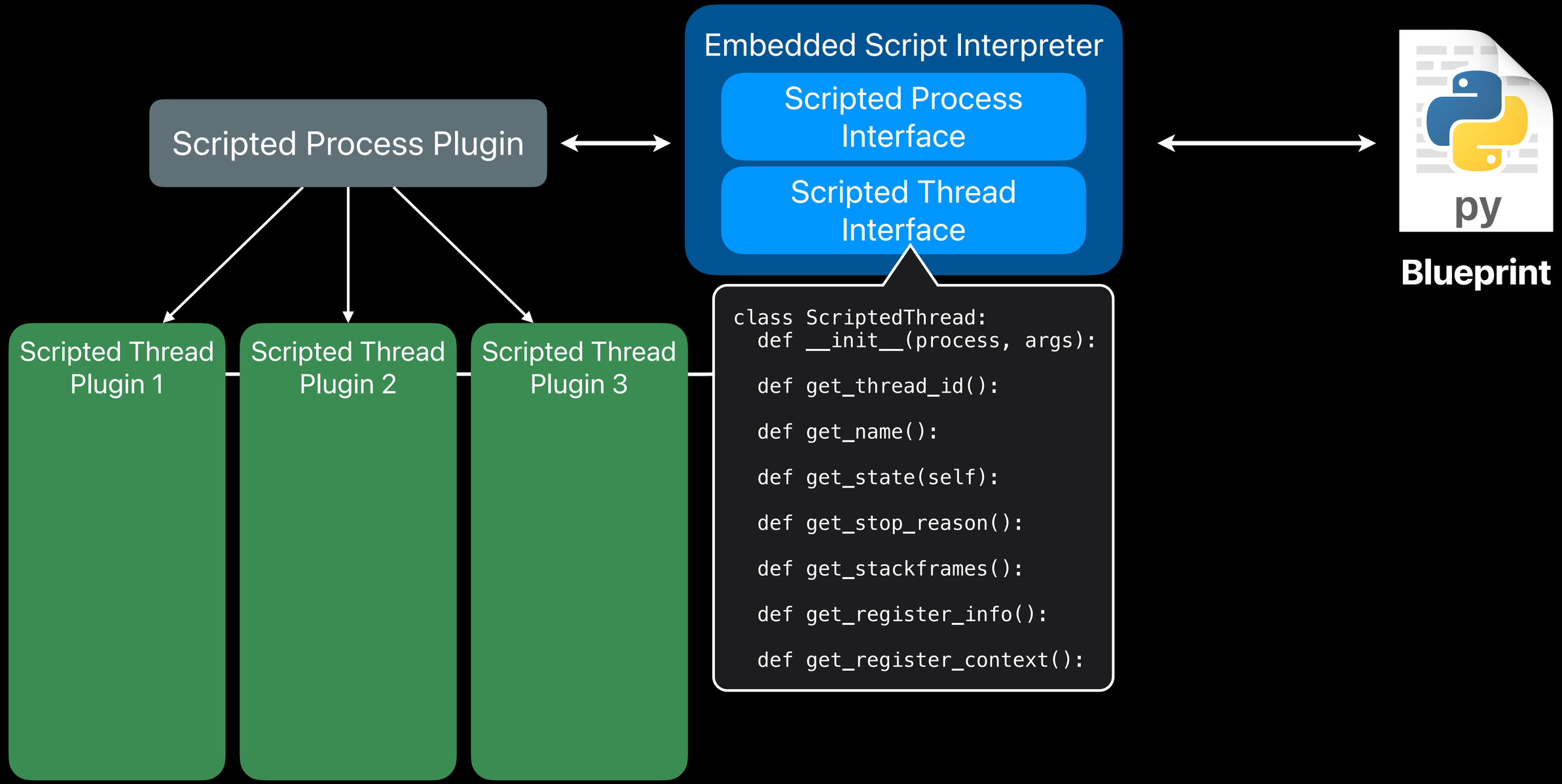


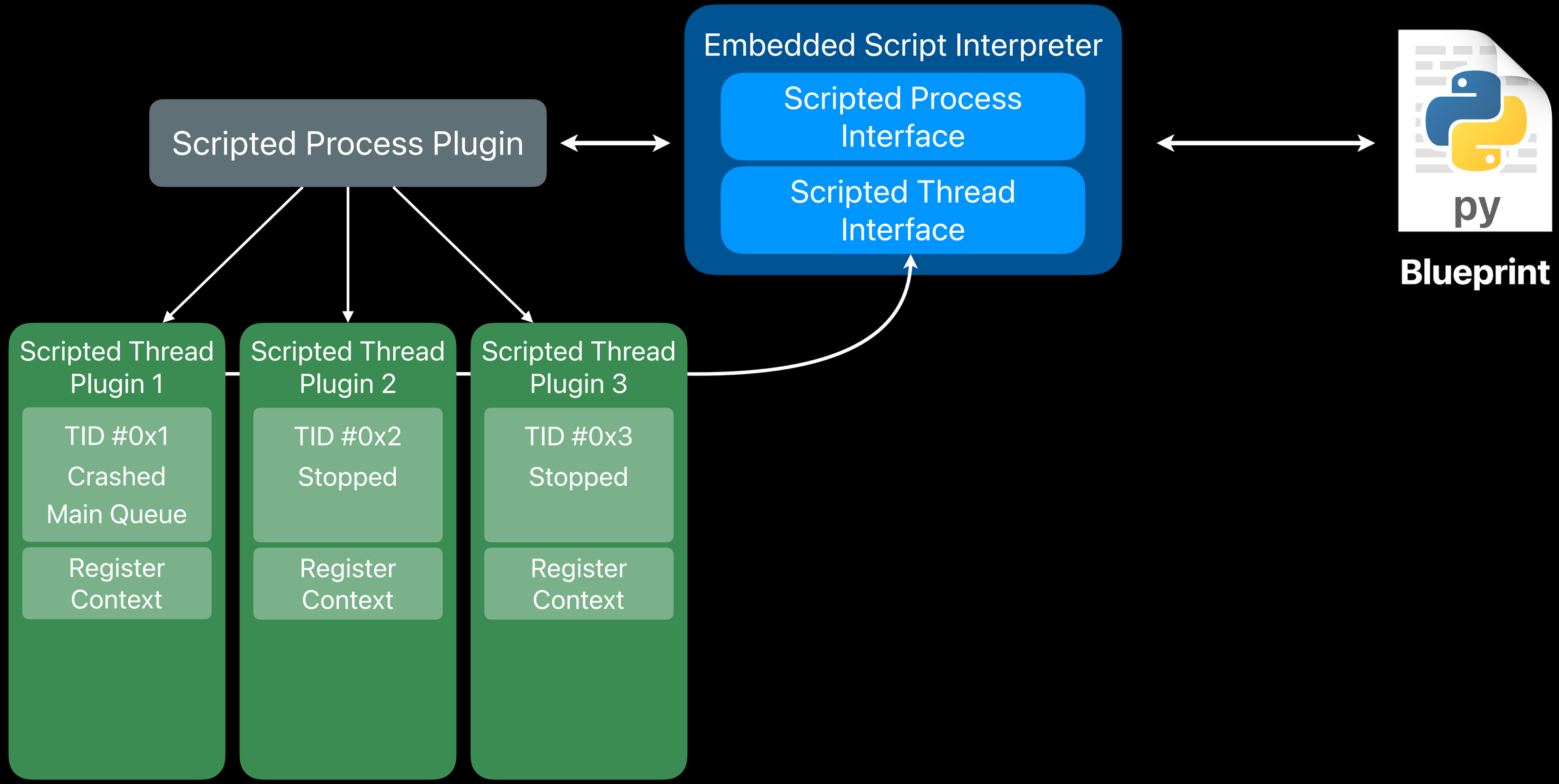

Scripted Process Plugin
Embedded Script Interpreter
Scripted Process Interface
py
Blueprint

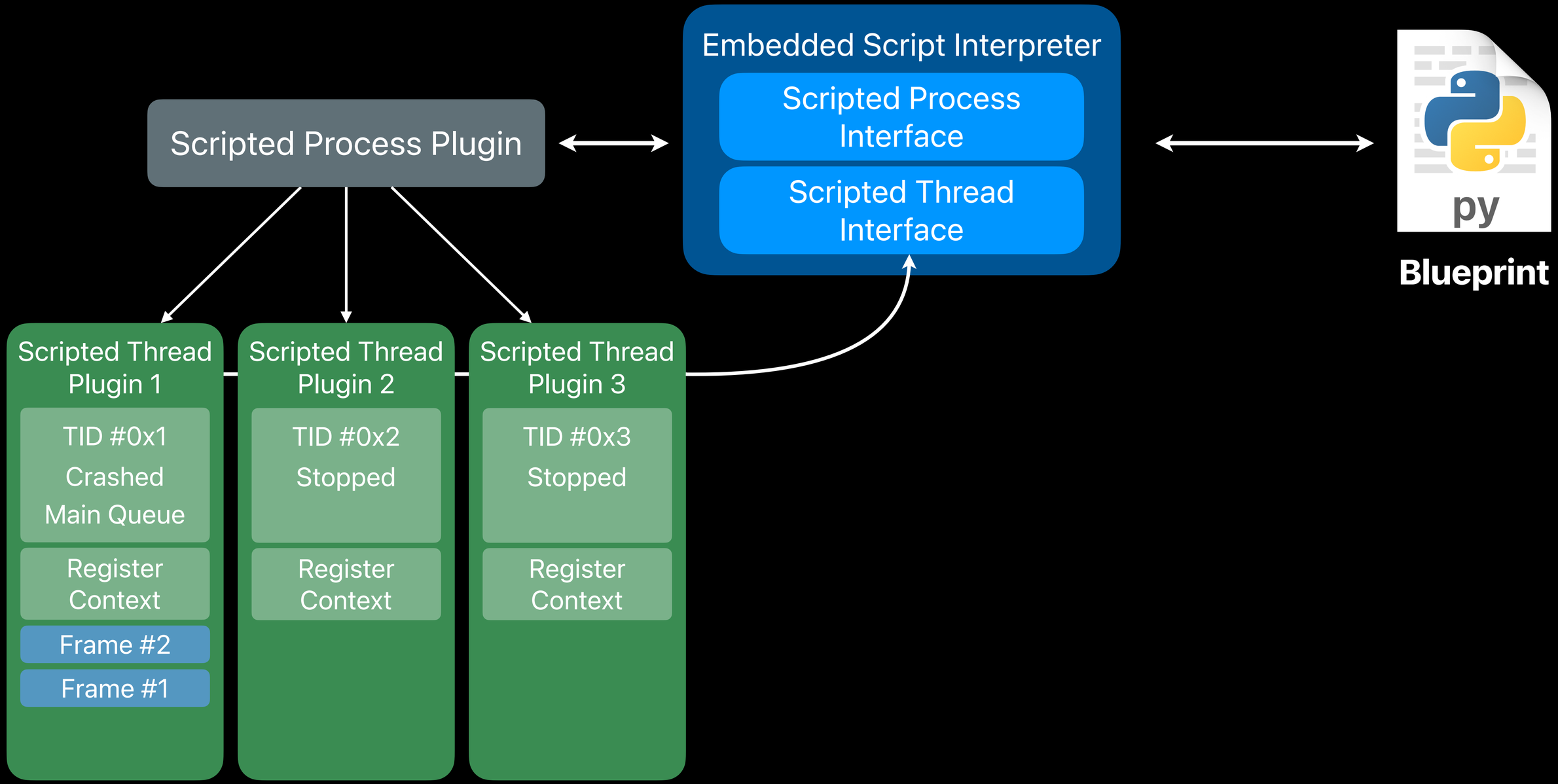

Scripted Process Plugin
Embedded Script Interpreter
Scripted Process Interface
py
Blueprint
Scripted Thread Plugin 1
Scripted Thread Plugin 2
Scripted Thread Plugin 3

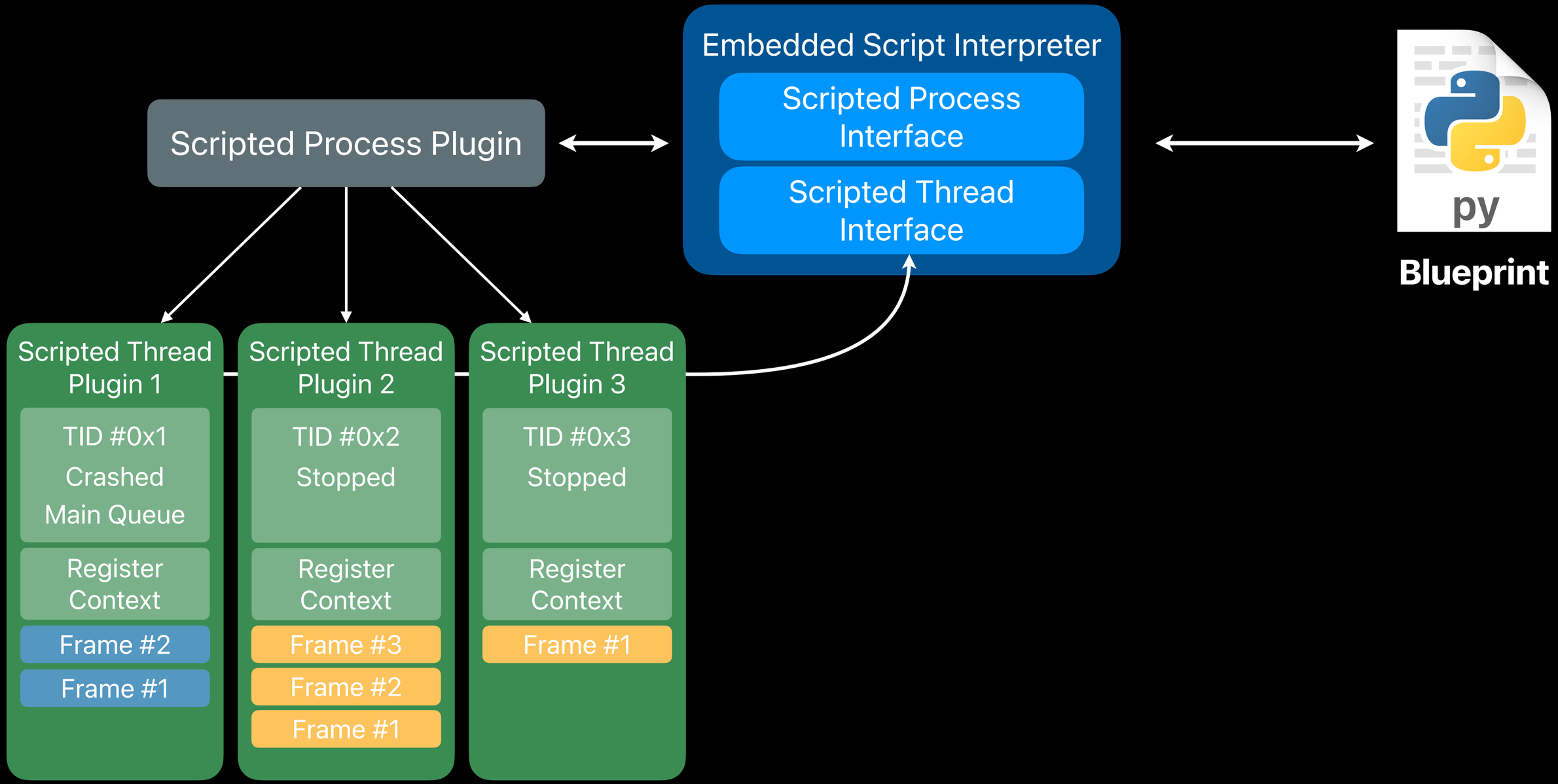

Embedded Script Interpreter
Scripted Process Interface
Scripted Thread Interface
Scripted Process Plugin
Scripted Thread Plugin 1
Scripted Thread Plugin 2
Scripted Thread Plugin 3
py
Blueprint


Embedded Script Interpreter
Scripted Process Interface
Scripted Thread Interface
Scripted Process Plugin
Scripted Thread Plugin 1
Scripted Thread Plugin 2
Scripted Thread Plugin 3
py
Blueprint


Embedded Script Interpreter
Scripted Process Interface
Scripted Thread Interface
Scripted Process Plugin
Scripted Thread Plugin 1
Scripted Thread Plugin 2
Scripted Thread Plugin 3
class ScriptedThread:
def __init__(process, args):
def get_thread_id():
def get_name():
def get_state(self):
def get_stop_reason():
def get_stackframes():
def get_register_info():
def get_register_context():
py
Blueprint


Embedded Script Interpreter
Scripted Process Interface
Scripted Thread Interface
Scripted Process Plugin
py
Blueprint
Scripted Thread Plugin 1
TID #0x1
Crashed
Main Queue
Register Context
Scripted Thread Plugin 2
TID #0x2
Stopped
Register Context
Scripted Thread Plugin 3
TID #0x3
Stopped
Register Context


Embedded Script Interpreter
Scripted Process Interface
Scripted Thread Interface
Scripted Process Plugin
py
Blueprint
Scripted Thread Plugin 1
TID #0x1
Crashed
Main Queue
Register Context
Frame #2
Frame #1
Scripted Thread Plugin 2
TID #0x2
Stopped
Register Context
Scripted Thread Plugin 3
TID #0x3
Stopped
Register Context


Embedded Script Interpreter
Scripted Process Interface
Scripted Thread Interface
Scripted Process Plugin
py
Blueprint
Scripted Thread Plugin 1
TID #0x1
Crashed
Main Queue
Register Context
Frame #2
Frame #1
Scripted Thread Plugin 2
TID #0x2
Stopped
Register Context
Frame #3
Frame #2
Frame #1
Scripted Thread Plugin 3
TID #0x3
Stopped
Register Context
Frame #1

The take-away

- Interactive Crashlogs provide a better way to investigate a crash
- Powered by Scripted Process, a new infrastructure for creating processes in lldb
- New opportunities

The take-away

- Interactive Crashlogs provide a better way to investigate a crash
- Powered by Scripted Process, a new infrastructure for creating processes in lldb
- New opportunities

The take-away

- Interactive Crashlogs provide a better way to investigate a crash

- Powered by Scripted Process, a new infrastructure for creating processes in lldb

- New opportunities

The take-away

- Interactive Crashlogs provide a better way to investigate a crash
- Powered by Scripted Process, a new infrastructure for creating processes in lldb
- New opportunities

The take-away

- Interactive Crashlogs provide a better way to investigate a crash
- Powered by Scripted Process, a new infrastructure for creating processes in lldb
- New opportunities

Thank you!