JSON output specification

When Scribble is executed with--output-mode json (or-m json), it instruments the target file and compiles the instrumented source. It then returns thestandard solc compilation output with 2 additions:

- In thesources
- field each file contains asource
- property with the source code of the instrumented file, along with the AST.
- A newinstrumentationMetadata
- field is added, which contains helpful metadata and instrumented-to-original source mapping information.
- You can find an example of the instrumentation metadata format in the next section.
-

Instrumentation metadata

Scribble provides metadata that allows you to explore in more detail how exactly the original source code was instrumented. This gives an insight on the correspondence of the annotations to the added code fragments in instrumented source.

Data structure format of the metadata is following:

```

Copy { // Instrumented-to-original source mapping. // Each entry is an array where // first element is the source range of instrumented source, // and second element is the corresponding source range of original source. "instrToOriginalMap": [ [ "61:532:0",// Source range in instrumented source "37:139:0"// Source range in original source ],

// ... may contain multiple entries ... ],

// Source ranges of additional instrumented code added by scribble. // I.e. this code doesn't directly correspond to un-instrumented code. "otherInstrumentation": [ "681:38:0",// Source range in instrumented source

// ... may contain multiple entries ... ],

// Information for each annotation declared in the orignal code "propertyMap": [ { // Unique Annotation id (assigned internaly by scribble). "id":1,

// Contract where annotation was defined "contract":"Test",

// Original file name where annotation was defined "filename":"sample.sol",

// The source range of the annotation expression in the original file. // For example in the annotation if_succeeds {:msg "P0"} y == x + 1; // the range would correspond to the y = x + 1 fragment. "propertySource":"85:11:0",

// The source range of a whole annotation in the original file. // For example in the annotation if_succeeds {:msg "P0"} y == x + 1; // the range would correspond to the entire if_succeeds {:msg "P0"} y == x + 1; fragment. "annotationSource":"61:36:0",

// Type of the annotation target (currently either function, contract, statement or state variable) "target":"function",

// Name of the annotation target. "targetName":"some",

// The signature (4 bytes in hex) for an additional debug event emitted when this annotation fails. "debugEventSignature":"",

// The human-readable label for this annotation. // For example in the annotation if_succeeds {:msg "P0"} y == x + 1; // this is the P0 string. "message":"P0",

// Array of source ranges of instrumentation code // that corresponding to this annotation. "instrumentationRanges":[ "251:29:0",

// ... may contain multiple entries ... ],

// Array of source ranges of instrumentation fragments // that correspond to the annotation predicate evaluation. "checkRanges":[ "221:14:0"

// ... may contain multiple entries ... ] } ],

// Array of the original source file names, // that have been instrumentated. "originalSourceList": [ "sample.sol" ],

// Array of the generated instrumentation file names. //"--" indicates we are outputing to STDOUT instead of files. "instrSourceList": [ "--" ] }

```

Example

Assuming we have asample.sol file with following Solidity code:

```

Copy pragma solidity ^0.8.0;

/// #define inc(uint x) uint = x + 1; contract Test { /// #if_succeeds {:msg "P0"} y == inc(x); function some(uint x) public returns (uint y) { y = x + 1; } }

```

If we run it with command:

```

Copy scribblesample.sol--output-modejson--output--

```

Scribble will produce and compile following instrumented source:

```

Copy pragma solidity 0.8.0;

/// #define inc(uint x) uint = x + 1; contract Test { event AssertionFailed(string message);

function some(uint x) public returns (uint y) { y = _original_Test_some(x); if (!(y == inc(x))) { emit AssertionFailed("1: P0"); assert(false); } }

function _original_Test_some(uint x) private returns (uint y) { y = x + 1; }

/// Implementation of user function define inc(uint256 x) uint256 = (x + 1) function inc(uint256 x1) internal view returns (uint256) { return x1 + 1; } } /// Utility contract holding a stack counter contract __scribble_ReentrancyUtils { bool __scribble_out_of_contract = true; }

```

And then will produce following JSON output:

```

Copy { "errors": [ // Common output of the Solc ], "sources": { // The flattened instrumented source, produced by Scribble: "flattened.sol":{ "ast":{ // Common output of the Solc }, "id":0,

// Generated instrumentation source: "source": "pragma solidity 0.8.0;\n\n/// define inc(uint x) uint = x + 1;\ncontract Test {\n event AssertionFailed(string message);\n\n function some(uint x) public returns (uint y) {\n y = _original_Test_some(x);\n if (!(y == inc(x))) {\n emit AssertionFailed(\"1: P0\");\n assert(false);\n }\n }\n\n function _original_Test_some(uint x) private returns (uint y) {\n y = x + 1;\n }\n\n /// Implementation of user function define inc(uint256 x) uint256 = (x + 1)\n function inc(uint256 x1) internal view returns (uint256) {\n return x1 + 1;\n }\n}\n/// Utility contract holding a stack counter\ncontract __scribble_ReentrancyUtils {\n bool __scribble_out_of_contract = true;\n}" } }, "contracts": { // Common output of the Solc }, // Instrumentation metadata, produced by Scribble: "instrumentationMetadata": { "instrToOriginalMap":[ [ "61:532:0", "62:139:0" ], [ "330:88:0", "127:72:0" ], [ "358:8:0", "140:8:0" ], [ "359:6:0", "141:6:0" ], [ "359:4:0", "141:4:0" ], [ "383:8:0", "164:8:0" ], [ "384:6:0", "165:6:0" ], [ "384:4:0", "165:4:0" ], [ "392:26:0", "173:26:0" ], [ "402:9:0", "183:9:0" ], [ "402:9:0", "183:9:0" ], [ "402:1:0", "183:1:0" ], [ "406:5:0", "187:5:0" ], [ "406:1:0", "187:1:0" ], [ "410:1:0", "191:1:0" ], [ "221:14:0", "86:36:0" ] ], "otherInstrumentation":[ "681:38:0", "571:13:0", "163:6:0", "181:26:0" ], "propertyMap":[ { "id":1, "contract":"Test", "filename":"sample.sol", "propertySource":"110:11:0", "annotationSource":"86:36:0", "target":"function", "targetName":"some", "debugEventSignature":"", "message":"P0", "instrumentationRanges":[ "251:29:0", "217:101:0" ], "checkRanges":[ "221:14:0" ] } ], "originalSourceList":[ "sample.sol" ], "instrSourceList":[ "--" ] } }

```

Previous Debugging Instrumented Code Next FAQ Last updated2 years ago

On this page * Instrumentation metadata * Example

Was this helpful?