# Intel® Stress Bitstreams and Encoder 2016 – Analytic Tools User Guide

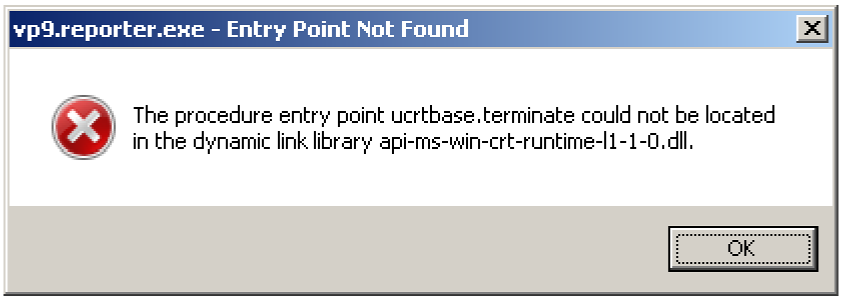
(version 2.1)

## Overview

Intel® Stress Bitstreams and Encoder 2016 — Analytic Tools package includes several tools to help decoder developers identify the right set of streams for testing purposes.

## Using Analytic Tools on old version of Windows

Analytic Tools require Visual Studio 2015 redistributable to be installed on target machine. If you get the following error message when running any tool please install redistributable from Microsoft* web site (https://www.microsoft.com/en-US/download/details.aspx?id=48145).



vp9.reporter.exe - Entry Point Not Found

The procedure entry point ucrtbase.terminate could not be located in the dynamic link library api-ms-win-crt-runtime-l1-1-0.dll.

## Limitations of Evaluation version

To use tools from evaluation version of Intel® SBE you need to copy valid evaluation license to the tools folder. Coverage Collector and Report Generator don't work without license. In addition, evaluation version of the Coverage Collector process only ten first frames of each stream and in case of HEVC tools picture size limited to 1920x1088.

## Coverage Collector

Coverage Collector is the core component of Analytic Tools. It is used to collect code and syntax element coverage for set of streams. The tool requires single command line argument — folder name with stream set to process. You can specify extensions of files to process using one or more `--mask` options. The tool collects information and creates coverage cache in the ***.coverages*** folder by default. Location of cache folder can be changed with `--cache` option. All other tools use data from cache so don't forget to use the same cache location for all tools. It is safe to run collector on the same set of streams multiple times because it checks if coverage for a stream already collected. Collector can process multiple streams in parallel. By default, number of parallel workers equal to number of CPU physical cores, but it can be changed with `--workers` command-line option. If you have multi-core CPU with hyper-threading, you can increase this number.

Please do not edit or remove any files in cache folder they are intended to be used by automatic tools.

To review available tool options run Coverage Collector with `--help` option:

```
C:\demo>hevc.collector.exe --help
Coverage Collector for HEVC 2.1 (part of the Intel(R) Stress Bitstreams and Encoder)
Copyright (c) 2015, Intel Corporation. All rights reserved.

usage: hevc.collector.exe [-h] [-m MASK] [--cache <cache-folder>]
                          [--workers WORKERS]
                          <folder>

positional arguments:
  <folder>              streams folder

optional arguments:
  -h, --help            show this help message and exit
```

*Other names and brands may be claimed as the property of others.

```
  -m MASK, --mask MASK  files to process (default: *.hevc)

others:
  --cache <cache-folder>
                        cache folder (default: .coverages)
  --workers WORKERS     number of streams processed in parallel (default: 2)
```

To collect coverage for all streams with ***bin*** and ***bit*** extensions from ***c:\streams*** folder with single worker use the following command line:

```
C:\demo>hevc.collector.exe c:\streams -m '*.bin' -m '*.bit' --workers 1
```

## Report Generator

Report Generator used to generate branch and syntax coverage report in HTML format. This report can be found in all Intel® SBE packages in the ***Branch_and_Syntax_Coverage_Static_View*** folder. To generate report you need to specify set of files to process using positional folder argument and optional `--mask` option. To customize report you can specify report folder (`--report`), title (`--title`) and sorting approach for source files and syntax elements (`--alpha`). By default, reporter continues processing of streams even if no coverage found in cache folder for some of them, this behavior can be overridden with `--fail-if-missing` option. Before report generation the tool check if destination folder already exists, but if you are ready to overwrite existing report, you can use `--overwrite` option. The most powerful option is `--coverage`, which allows setting coverage criteria for the report. Format of this criteria will be described in the following chapter. Many other tools can use coverage criteria to customize their results.

```
Coverage Report Generator for HEVC 2.1 (part of the Intel(R) Stress Bitstreams and Encoder)
Copyright (c) 2015, Intel Corporation. All rights reserved.

usage: hevc.reporter.exe [-h] [-m MASK] [-r <report-folder>] [-t TITLE]
                         [--overwrite] [--fail-if-missing] [-c <yaml>]
                         [--alpha] [--cache <cache-folder>]
                         <folder>

positional arguments:
  <folder>              streams folder

optional arguments:
  -h, --help            show this help message and exit
  -m MASK, --mask MASK  files to process (default: *.hevc)

report:
  -r <report-folder>, --report <report-folder>
                        folder name to save report (default: basecov)
  -t TITLE, --title TITLE
                        report title (default: Branch and Syntax Coverage
                        Report)
  --overwrite           overwrite existing report
  --fail-if-missing     fail if no cached coverage for any stream
  -c <yaml>, --coverage <yaml>
                        custom coverage criteria in YAML format
  --alpha               sort syntax elements and source file alphabetically

others:
  --cache <cache-folder>
                        cache folder (default: .coverages)
```

To generate report in the ***custom_report*** folder for the same set of streams and coverage criteria from the package run the following command-line:

```
C:\demo>hevc.reporter.exe c:\streams -m '*.bin' -m '*.bit' --report custom_report --coverage hevc.criteria.yaml
```

Home page of the report is **basecov.html** file in the destination folder i.e. ***custom_report*** in our case.

## Coverage Criteria

It is file in YAML format, which can be used by several tools. It is possible to exclude syntax element from results using the following syntax:

```
element:
  skip: yes
```

The most important use case is definition of coverage criteria for an element. If not all values of syntax element are important, you can specify list of important values using:

```
element:
  coverage:
    key-values: [value1, value2,…, valueN]
```

If individual element values are not important you can define either number of values or percentage of values to cover:

```
element:
  coverage:
    values: N
```

or

```
element:
  coverage:
    percentage: M
```

It is possible to combine list of important values and number of extra values to cover using either “values” or “percentage” keys:

```
element:
  coverage:
    key-values: [0, 100]
    values: 5
```

In this case, to get 100% coverage streams should cover values 0, 100, and five other values.

```
element:
  coverage:
    key-values: [1, 2, 3]
    percentage: 10
```

In this example, ten percent of all values should be covered including values 1, 2, and 3.

## Stream Pool Optimizer

Optimizer used to find subset of streams with the same coverage as the whole stream pool. There are two usage scenarios: finding minimal subset of streams for smoke testing and finding subset of streams to test particular list of functions and/or syntax elements.

The tool uses the following command-line options:

```
C:\demo>hevc.optimizer.exe --help
Stream Pool Optimizer for HEVC 2.1 (part of the Intel(R) Stress Bitstreams and Encoder)
Copyright (c) 2015, Intel Corporation. All rights reserved.

usage: hevc.optimizer.exe [-h] [-m MASK] [--fail-if-missing] [-c <yaml>]
                          [--functions <function-list>]
                          [--elements <syntax-element-list>] [--copy <dest>]
                          [--cache <cache-folder>]
                          <folder>

positional arguments:
  <folder>              streams folder

optional arguments:
  -h, --help            show this help message and exit
  -m MASK, --mask MASK  files to process (default: *.hevc)
  --fail-if-missing     fail if no cached coverage for any stream
  -c <yaml>, --coverage <yaml>
                        custom coverage criteria in YAML format
  --functions <function-list>
                        file with list of function-of-interest
  --elements <syntax-element-list>
                        file with list of element-of-interest
  --copy <dest>         copy selected streams to <dest>
  --cache <cache-folder>
                        cache folder (default: .coverages)
```

Streams folder, file masks, coverage criteria, cache folder and behavior for streams missed in the cache already described in previous chapters. Let's review new options.

Functions of interest (`--functions`) option defines name of file with list of functions which coverage is important for you. If you don't use this option, the tool finds streams to cover all functions. Similar option `--elements` used to define file name with list of syntax elements. If you are interested in code or syntax coverage only, you can point any of these options to empty file. The tool check both lists to ensure all defined functions and elements are known to prevent unexpected results. Optimizer does not allows using empty files for both options at once.

The final option is `--copy`. You can use it to define destination folder to copy optimized list of files. If it does not exist, the tool will create it for you.

For example:

```
C:\demo>hevc.optimizer.exe c:\streams --coverage hevc.criteria.yaml --copy c:\optimized
```

As a result minimal set of streams with the same coverage as all streams (by default the tool analyze all files with .hevc extension) from ***c:\streams*** folder will be copied to ***c:\optimized*** directory.

## Coverage Difference

To compare coverages for two set of streams you use this tool, which highlights difference for individual syntax element values and basic blocks.

```
C:\demo>hevc.differ.exe --help
Coverage Difference for HEVC 2.1 (part of the Intel(R) Stress Bitstreams and Encoder)
Copyright (c) 2015, Intel Corporation. All rights reserved.

usage: hevc.differ.exe [-h] [-m MASK] [--fail-if-missing] [-c <yaml>]
                       [--base-cache <base-cache-folder>]
                       [--new-cache <new-cache-folder>] [--report REPORT]
                       [--base-title <base-title>] [--new-title <new-title>]
                       <base> <new>

positional arguments:
  <base>                base streams folder
  <new>                 new streams folder

optional arguments:
  -h, --help            show this help message and exit
  -m MASK, --mask MASK  files to process (default: *.hevc)
  --fail-if-missing     fail if no cached coverage for any stream
  -c <yaml>, --coverage <yaml>
                        custom coverage criteria in YAML format
  --base-cache <base-cache-folder>
                        cache folder (default: .coverages)
  --new-cache <new-cache-folder>
```

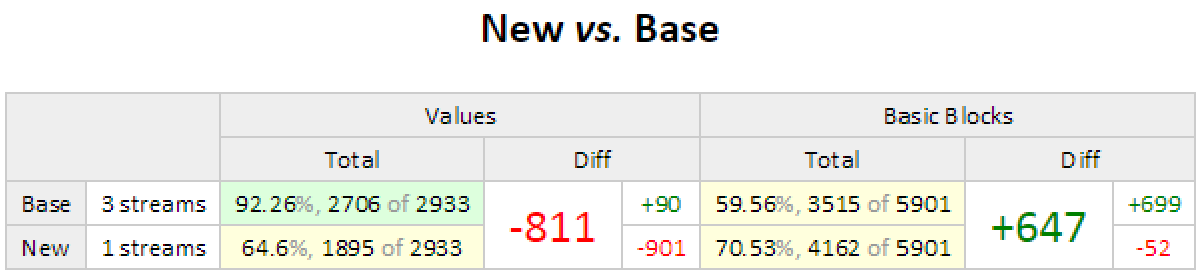
```
                         cache folder (default: .coverages)

report:
  --report REPORT        report file name
  --base-title <base-title>
                         title for the base streams (default: Base)
  --new-title <new-title>
                         title for the new streams (default: New)
```

As for regular diff tool, you need to specify left (base) and right (new) parts for comparison. Minimum you need to define two folders with streams. In addition you can specify cache folders (`--base-cache` and `--new-cache`), titles (`--base-title` and `--new-title`) and file name for the generated report (`--report`). All other options have the same meaning as in all previous tools.

If you run coverage difference collection for full set of streams and optimized subset, you will see that there are no differences.

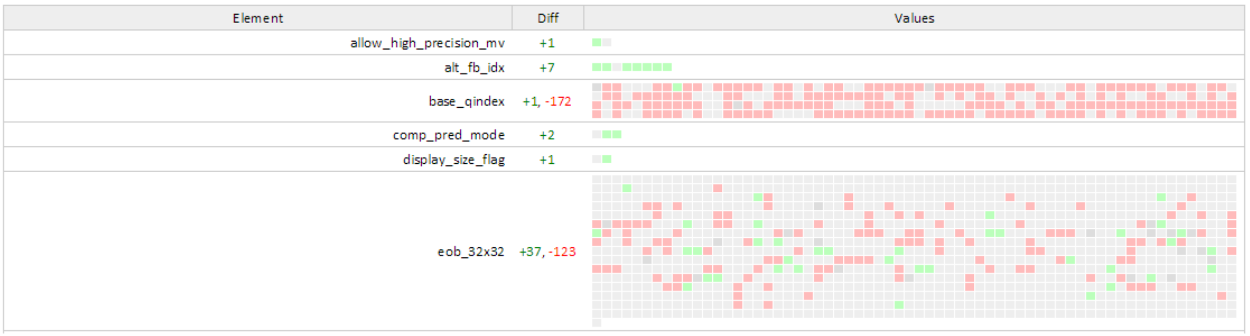
In other cases you will see html report with three sections. High level summary:



**New vs. Base**

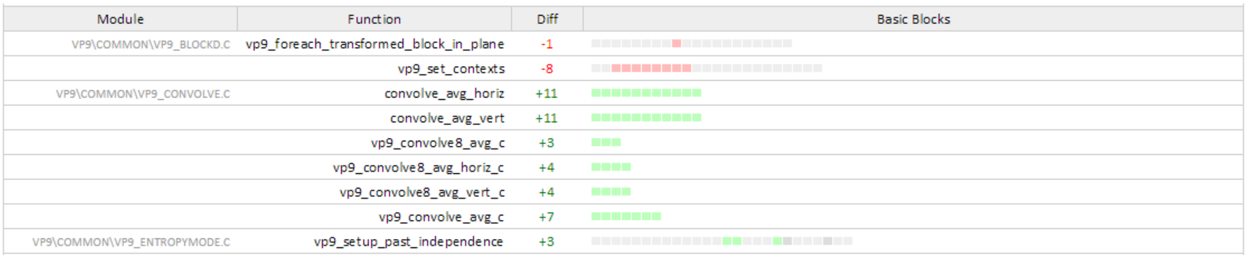
| | | Values | | | Basic Blocks | | |
|---|---|---|---|---|---|---|---|
| | | Total | Diff | | Total | Diff | |
| Base | 3 streams | 92.26%, 2706 of 2933 | -811 | +90 | 59.56%, 3515 of 5901 | +647 | +699 |
| New | 1 streams | 64.6%, 1895 of 2933 | | -901 | 70.53%, 4162 of 5901 | | -52 |

Syntax Element Values difference:



| Element | Diff | Values |
|---|---|---|
| allow_high_precision_mv | +1 | |
| alt_fb_idx | +7 | |
| base_qindex | +1, -172 | |
| comp_pred_mode | +2 | |
| display_size_flag | +1 | |
| eob_32x32 | +37, -123 | |

and Basic Block difference for each function:



| Module | Function | Diff | Basic Blocks |
|---|---|---|---|
| VP9\COMMON\VP9_BLOCKD.C | vp9_foreach_transformed_block_in_plane | -1 | |
| | vp9_set_contexts | -8 | |
| VP9\COMMON\VP9_CONVOLVE.C | convolve_avg_horiz | +11 | |
| | convolve_avg_vert | +11 | |
| | vp9_convolve8_avg_c | +3 | |
| | vp9_convolve8_avg_horiz_c | +4 | |
| | vp9_convolve8_avg_vert_c | +4 | |
| | vp9_convolve_avg_c | +7 | |
| VP9\COMMON\VP9_ENTROPYMODE.C | vp9_setup_past_independence | +3 | |

Low level differences show number of added and removed values or basic blocks with color-coded tables where each cell represents single value or basic block. Green cells indicate values added in new stream set and red ones indicate removed values. If individual value or basic block covered or uncovered by both sets of streams.

## Stream Analyzer

If you have long stream, vp9.analyzer can find minimal stream length with the same level of coverage as full stream has. This tool available only for VP9 streams.

```
C:\demo>vp9.analyzer.exe --help
Stream Coverage Analyzer for VP9 2.1 (part of the Intel(R) Stress Bitstreams and Encoder)
Copyright (c) 2015, Intel Corporation. All rights reserved.

usage: vp9.analyzer.exe [-h] [-c <yaml>] [--functions <function-list>]
                        [--elements <syntax-element-list>]
                        <stream>

positional arguments:
  <stream>              stream to analyze

optional arguments:
  -h, --help            show this help message and exit
  -c <yaml>, --coverage <yaml>
                        custom coverage criteria in YAML format
  --functions <function-list>
                        file with list of function-of-interest
  --elements <syntax-element-list>
                        file with list of element-of-interest
```

The tool accepts file name to analyze as single positional arguments and `--coverage`, `--functions`, and `--elements` options to define coverage criteria and list of functions and/or syntax elements of interest.