CodeQL CLI SARIF output

In this article

You can output SARIF from the CodeQL CLI and share static analysis results with other systems.

GitHub CodeQL is licensed on a per-user basis upon installation. You can use CodeQL only for certain tasks under the license restrictions. For more information, see "About the CodeQL CLI." If you have a GitHub Advanced Security license, you can use CodeQL for automated analysis, continuous integration, and continuous delivery. For more information, see "About GitHub Advanced Security."

About SARIF output

SARIF is designed to represent the output of a broad range of static analysis tools, and there are many features in the SARIF specification that are considered "optional". This document details the output produced when using the format type `sarifv2.1.0`, which corresponds to the SARIF v2.1.0.csd1 specification. For more information on selecting a file format for your analysis results, see "database analyze."

SARIF specification and schema

This article is intended to be read alongside the detailed SARIF specification. For more information on the specification and the SARIF schema, see the SARIF specification documentation.

Change notes

Changes between versions

CodeQL version	Format type	Changes
2.0.0	`sarifv2.1.0`	First version of this format.

Future changes to the output

The output produced for a given specific format type (for example, `sarifv2.1.0`) may change in future CodeQL releases. We will endeavor to maintain backwards compatibility with consumers of the generated SARIF by ensuring that:

- Fields that are marked as always being generated will never be removed.
- For fields that are marked as not always being generated, the circumstances under which the fields are generated may change. Consumers of the CodeQL SARIF output should be robust to the presence or absence of these fields.

New output fields may be added in future releases under the same format type–these are not considered to break backwards compatibility, and consumers should be robust to the presence of newly added fields.

New format argument types may be added in future versions of CodeQL—for example, to support new versions of SARIF. These have no guarantee of backwards compatibility, unless explicitly documented.

Generated SARIF objects

This details each SARIF component that may be generated, along with any specific circumstances. We omit any properties that are never generated.

`sarifLog` object

JSON property name	Always generated?	Notes
`$schema`	✓	Provides a link to the SARIF schema.
`version`	✓	The version of the SARIF used to generate the output.
`runs`	✓	An array containing a single run object, for one language.

`run` object

JSON property name	Always generated?	Notes
`tool`	✓	None
`artifacts`	✓	An array containing at least one artifact object for every file referenced in a result.
`results`	✓	None
`newLineSequences`	✓	None
`columnKind`	✓	None
`properties`	✓	The properties dictionary will contain the `semmle.formatSpecifier`, which identifies the format specifier passed to the CodeQL CLI.

`tool` object

JSON property name	Always generated?	Notes

`driver`	✓	None

`toolComponent` object

JSON property name	Always generated?	Notes
`name`	✓	Set to "CodeQL command-line toolchain" for output from the CodeQL CLI tools. Note, if the output was generated using a different tool a different `name` is reported, and the format may not be as described here.
`organization`	✓	Set to "GitHub".
`version`	✓	Set to the CodeQL release version e.g. "2.0.0".
`rules`	✓	An array of `reportingDescriptor` objects that represent rules. This array will contain, at a minimum, all the rules that were run during this analysis, but may contain rules which were available but not run. For more detail about enabling queries, see `defaultConfiguration`.

`reportingDescriptor` object (for rule)

`reportingDescriptor` objects may be used in multiple places in the SARIF specification. When a `reportingDescriptor` is included in the rules array of a `toolComponent` object it has the following properties.

JSON property name	Always generated?	Notes
`id`	✓	Will contain the `@id` property specified in the query that defines the rule, which is usually of the format `language/rule-name` (for example `cpp/unsafe-format-string`). If your organization defines the `@opaqueid` property in the query it will be used instead.
`name`	✓	Will contain the `@id` property specified in the query. See the `id` property above for an example.
`shortDescription`	✓	Will contain the `@name` property specified in the query that defines the rule.
`fullDescription`	✓	Will contain the `@description` property specified in the query that defines the rule.

`defaultConfiguration`	✓	A `reportingConfiguration` object, with the enabled property set to true or false, and a level property set according to the `@severity` property specified in the query that defines the rule. Omitted if the `@severity` property was not specified.

`artifact` object

JSON property name	Always generated?	Notes
`location`	✓	An `artifactLocation` object.
`index`	✓	The index of the `artifact` object.
`contents`	×	If results are generated using the `--sarif-add-file-contents` flag, and the source code is available at the time the SARIF file is generated, then the `contents` property is populated with an `artifactContent` object, with the `text` property set.

`artifactLocation` object

JSON property name	Always generated?	Notes
`uri`	✓	None
`index`	✓	None
`uriBaseId`	×	If the file is relative to some known abstract location, such as the root source location on the analysis machine, this will be set.

`result` object

The composition of the results is dependent on the options provided to CodeQL. By default, the results are grouped by unique message format string and primary location. Thus, two results that occur at the same location with the same underlying message, will appear as a single result in the output. This behavior can be disabled by using the flag `--ungroup-results`, in which case no results are grouped.

JSON property name	Always generated?	Notes
`ruleId`	✓	See the description of the `id` property in `reportingDescriptor` object (for rule) .
`ruleIndex`	✓	None

`message`	✓	A message describing the problem(s) occurring at this location. This message may be a SARIF "Message with placeholder", containing links that refer to locations in the `relatedLocations` property.
`locations`	✓	An array containing a single `location` object.
`partialFingerprints`	✓	A dictionary from named fingerprint types to the fingerprint. This will contain, at a minimum, a value for the `primaryLocationLineHash`, which provides a fingerprint based on the context of the primary location.
`codeFlows`	✗	This array may be populated with one or more `codeFlow` objects if the query that defines the rule for this result is of `@kind path-problem`.
`relatedLocations`	✗	This array will be populated if the query that defines the rule for this result has a message with placeholder options. Each unique location is included once.
`suppressions`	✗	If the result is suppressed, then this will contain a single `suppression` object, with the `@kind` property set to `IN_SOURCE`. If this result is not suppressed, but there is at least one result that has a suppression, then this will be set to an empty array, otherwise it will not be set.

`location` object

JSON property name	Always generated?	Notes
`physicalLocation`	✓	None
`id`	✗	`location` objects that appear in the `relatedLocations` array of a `result` object may contain the `id` property.
`message`	✗	`location` objects may contain the `message` property if: - They appear in the `relatedLocations` array of a `result` object may contain the `message` property.

		- They appear in the `threadFlowLocation.location` property.

`physicalLocation` object

JSON property name	Always generated?	Notes
`artifactLocation`	✓	None
`region`	×	If the given `physicalLocation` exists in a text file, such as a source code file, then the `region` property may be present.
`contextRegion`	×	May be present if this location has an associated `snippet`.

`region` object

There are two types of `region` object produced by CodeQL:

- Line/column offset regions
- Character offset and length regions

Any region produced by CodeQL may be specified in either format, and consumers should robustly handle either type.

For line/column offset regions, the following properties will be set:

JSON property name	Always generated?	Notes
`startLine`	✓	None
`startColumn`	×	Not included if equal to the default value of 1.
`endLine`	×	Not included if identical to `startLine`.
`endColumn`	✓	None
`snippet`	×	None

For character offset and length regions, the following properties will be set:

JSON property name	Always generated?	Notes
`charOffset`	×	Provided if `startLine`, `startColumn`, `endLine`, and `endColumn` are not populated.
`charLength`	×	Provided if `startLine`, `startColumn`, `endLine`, and `endColumn` are not populated.
`snippet`	×	None

`codeFlow` object

JSON property name	Always generated?	Notes
`threadFlows`	✓	None

`threadFlow` object

JSON property name	Always generated?	Notes
`locations`	✓	None

`threadFlowLocation` object

JSON property name	Always generated?	Notes
`location`	✓	None