WARNING SEARCH LANGUAGE

The CodeSonar *warning search language* allows you to specify search conditions based on various warning properties, and provides basic logical operators for combining multiple conditions. It can be used both to specify and to refine searches.

- **Use with the simple search tool to specify a search in the warning domain.**
    - **With the warnings in these results domain/scope setting on the warning search results page, will narrow the results of a previous search.**
- **Use in the chart wizard search tab to specify which warnings should be charted.**
- **Use to interpret the warning searches described on the Saved Searches page.**

Warning-Specific Grammar

The warning search language grammar is based on the standard CodeSonar search grammar, with several extensions:

- In addition to the `T` types provided in the standard search grammar, there is a new `T set-operator T` term.
    - **Important Note:** Any search that includes one or more `T set-operator T` terms will return a result set containing at most one instance per warning group. This is the case even if Visible Warnings is set to **all**.
    - Precedence rules are as follows.

      Within `T`:

`()` > unary > binary > set-operator

Within set-operator:

INTERSECT > UNION = DIFFERENCE

(Within binary, precedence is still & > |)

- In addition to the condition types provided in the standard search grammar, there are warning-id-range and hash-value conditions.
- There are domain-specific field-name values.
- none (case insensitive) is not treated as an ordinary word. Instead, it matches *all* of the following.
    - literal string none (case insensitive)
    - fields whose value is the empty string
    - empty fields (that is, fields that don't contain any value at all, not even a default value)

  Plain text search in the warning search language covers a large number of fields, including some that many users choose to leave empty, such as Analysis Description (adesc). Consequently, performing a plain text search for none (or "none", which CodeSonar will normalize to none) may match many warnings that do not contain literal string none.To avoid this issue, use one or more field-condition, ilike-condition, or imatch-condition terms to restrict the search to the precise field or fields you are interested in.

Literal strings are displayed in pink text.

```
T : quoted-string
  | field-condition
  | ilike-condition
  | imatch-condition
  | ( T )
  | unary T
  | T binary T
  | T set-operator T
  | T T
  | word
  | none

field-name : adesc
           | aid
           | analysis
           | categories
           | class
```

```
            | cluster
            | clustered
            | detected
            | file
            | finding
            | fingerprint
            | firstdetected
            | id
            | iid
            | language
            | line
            | line_content
            | line_content_xml
            | listing
            | listing_xml
            | modified
            | new_warning
            | notes
            | owner
            | path
            | priority
            | project
            | pdesc
            | procedure
            | ptree_path
            | rank
            | score
            | significance
            | similar
            | sql
            | state

condition : warning-id-range
          | word
          | quoted-string
          | num-range
          | int-range
          | hash-value
          | date-range
          | boolean

set-operator : UNION
             | INTERSECT
             | DIFFERENCE

warning-id-range : warning-id
                 | int-range
                 | group-range
                 | both-range

warning-id : int
           | warning-id-grouponly
           | warning-id-instonly
           | warning-id-both

group-range : warning-id-grouponly..
            | ..warning-id-grouponly
```

```
            | warning-id-grouponly..warning-id-grouponly

both-range : warning-id-both..
           | ..warning-id-both
           | warning-id-both..warning-id-both

warning-id-grouponly : int.

warning-id-instonly : .int

warning-id-both: int.int

hash-value: hexadecimal number
```

`field-condition` terms

The following table shows the relationship between the warning-specific `field-name` values and the properties of a warning, along with the expected type of the `condition` part of a `field-condition`.

field-name	Contents	Condition Type
`adesc`	Warning Analysis . Analysis Description	`word` | `quoted-string`, or special value `none` (`*`).
`aid`	Warning Analysis . Analysis ID	`int-range`, or special value `last`.
`analysis`	Warning Analysis . Analysis	`word` | `quoted-string`, or special value `none` (`*`).
`categories`	Warning Class . Categories	`word` | `quoted-string`, or special value `none` (`*`).

class	Warning Class . Name	word | quoted-string, or special value none (*).
cluster	Cluster	hash-value
clustered	Cluster Representative?	boolean (*)
detected	Detected	date-range
directory	Warning File . Directory	word | quoted-string, or special value none (*).
file	Warning File . File	word | quoted-string, or special value none (*).
finding	Finding	word | quoted-string, or special value none (*).
fingerprint	Fingerprint	hash-value
firstdetected	First Detected	date-range
id	Warning ID	warning-id-range (*)
iid	Instance ID	int-range

language	Warning File . Language	word | quoted-string
line	Line Number	int-range
line_content	Line Content	word | quoted-string, or special value none (*).
line_content_xml	Line Content XML	word | quoted-string, or special value none (*).
listing	Listing	word | quoted-string, or special value none (*).
listing_xml	Listing XML	word | quoted-string, or special value none (*).
modified	Modified	date-range
new_warning	"true" if no warning instance on the hub has the same Group ID but an earlier Warning Analysis (as determined by Analysis ID), "false" otherwise.	boolean
notes	User Notes	word | quoted-string, or special value none (*).

owner	Owner	word | quoted-string, or special value none (*).
path	Warning File . File Path	word | quoted-string, or special value none (*).
pdesc	Warning Project . Project Description	word | quoted-string, or special value none (*).
pid	Warning Project . Project ID	int-range
priority	Priority	word | quoted-string, or special value none (*).
procedure	Procedure	word | quoted-string, or special value none (*).
project	Warning Project . Name	word | quoted-string, or special value none (*).
ptree_path	Warning Project . Project Path	word | quoted-string, or special value none (*).
rank	Rank	int-range (*)

score	Score	int-range (*)
significance	Warning Class . Significance	word | quoted-string, or special value none (*).
similar	Representative?	boolean (*)
state	State	word | quoted-string, or special value none (*).
sql	Introduces an SQL expression (see SQL Terms, below).	word | quoted-string

Terms involving a warning-id-range

The following table shows the different forms an id-based field-condition term can take, along with their interpretations.

id:*X*	Group ID is *X* or Instance ID is *X*.
id:*A*.	Group ID is *A*.
id:*A*..	Group ID is *A* or higher
id:..*C*	Group ID is *C* or lower
id:*A*..*C*	Group ID is between *A* and *C*, inclusive.
id:.*B*	Instance ID is *B*.
id:.*B*..	Instance ID is *B* or higher.
id:...*B*	Instance ID is *B* or lower.
id:.*B*...*D*	Instance ID is between *B* and *D*, inclusive.
id:*A*.*B*	Group ID is *A* and instance id is *B*

`id:A.B..`	Group ID is *A* or higher. If Group ID is *A*, Instance ID is *B* or higher.
`id:..A.B`	Group ID is *A* or lower. If Group ID is *A*, Instance ID is *B* or lower.
`id:A.B..C.D`	Group ID is between *A* and *C*, inclusive. If Group ID is *A*, Instance ID is *B* or higher. If Group ID is *C*, Instance ID is *D* or lower.

`similar` and `clustered` searches

A `field-condition` term with `field-name` `similar` or `clustered` is interpreted as follows.

`clustered:true`	All instances whose Cluster Representative? value is `true`.
`clustered:false`	All instances. (That is, all instances whose Cluster Representative? value is `true` AND all instances whose Cluster Representative? value is `false`.)
`similar:true`	All instances (That is, all instances whose Representative? value is `true` AND all instances whose Representative? value is `false`.)
`similar:false`	All instances whose Representative? value is `true`.

Terms involving a `set-operator`

Terms that involve a `set-operator` define a set of *candidate matches*, as described in the following table.

Term	Candidate Matches
T1 `UNION` *T2* *T1* `+` *T2*	The set of all warning instances `G.i` such that `G.i` matches *T1* **or** `G.i` matches *T2*.
T1 `INTERSECT` *T2* *T1* `*` *T2*	The set of all warning instances `G.i` such that: • `G.i` matches *T1* **and** there exists some instance of *G* that matches *T2*, **or** • `G.i` matches *T2* **and** there exists some instance of *G* that matches *T1*.

T1 `DIFFERENCE` *T2* *T1* `-` *T2*	The set of all warning instances *G*`.`*i* such that *G*`.`*i* matches *T1* **and** no instance of *G* matches *T2*.

Important Note: Any search that includes one or more `T set-operator T` terms will return a result set containing at most one instance per warning group. If two or more instances of a single warning group match the search query, scope, and Visible Warnings setting, one is selected arbitrarily for inclusion in the result set.

- With **active** Visible Warnings (more generally, any query or Visible Warnings setting that specifies `similar=false`), the result set will only contain representative instances.
- With **all** Visible Warnings (more generally, any search where neither query nor Visible Warnings specifies `similar=false`), the result set may contain non-representative instances - even if representative instances of the same warning group were eligible for inclusion.
- Note that this property means that the results for query *T1* `UNION` *T2* are a subset of the results for *T1* `OR` *T2*, and often a *strict* subset.

`directory`, `file`, and `path` searches

If `field-name` is `directory`, `file`, or `path`, comparison:

- is case-insensitive, and
- does not distinguish between forward slashes and backslashes.

`rank` and `score` terms

The rank and score for a warning instance are computed and stored as floating point values, although they are rounded before displaying in the GUI. The `int-range` conditions for `rank`- and `score`-based `field-condition` terms are therefore interpreted as follows.

- An isolated integer X matches all values in the interval [X-0.5, X+0.5), or [0, X+0.5).
- An integer X at the bottom of a range X.. or X..Y will be interpreted as X-0.5, or 0.
- An integer X at the top of a range W..X or ..X will be interpreted as X+0.5.

If a value lies outside the interval [0, 100] after this adjustment, it is further adjusted to the appropriate extreme of that interval.

For example:

search term	matches warnings where
`score=40`	39.5≤score<40.5
`score=30..40`	29.5≤score<40.5
`score=0..40`	0≤score<40.5
`score=0..140`	0≤score<100

Plain text terms: `word`, `quoted-string`, and `none`

Plain text search terms—word, quoted-string, and `none`—are compared against the following fields, with substring search.

- *`str`* is equivalent to `adesc:`*`str`* `| aid:`*`str`* `| analysis:`*`str`* `| categories:`*`str`* `| class:`*`str`* `| detected:`*`str`* `| directory:`*`str`* `| file:`*`str`* `| finding:`*`str`* `| fingerprint:`*`str`* `| firstdetected:`*`str`* `| id:`*`str`* `| iid:`*`str`* `| language:`*`str`* `| line:`*`str`* `| line_content:`*`str`* `| listing:`*`str`* `| modified:`*`str`* `| notes:`*`str`* `| owner:`*`str`* `| path:`*`str`* `| priority:`*`str`* `| procedure:`*`str`* `| rank:`*`str`* `| score:`*`str`* `| state:`*`str`*

Examples

One good source of example search language expressions is the Saved Searches page, since it lists the expression associated with each named search. Even if you haven't saved any searches yet, the page will list the expressions for **active** and **not suppressed**, which come with CodeSonar:

named search	expression and interpretation

not suppressed	`-priority=Suppressed similar=false clustered=true` This expression is interpreted as "find cluster representatives for warnings that do not have Priority equal to **Suppressed**". • `priority` on the left-hand side of an expression refers to the Priority field of a warning group. • = specifies an exact match. Any warnings with Priority set to **This Should Be Suppressed** (if such a priority existed) will be returned by this search. • `Suppressed` on the right-hand side of an expression is interpreted as an ordinary string. It doesn't need quote marks because it doesn't contain whitespace or special characters. • `-` negates `priority=Suppressed`. • `similar=false` matches only representative instances (all other instances have `similar=true`). • `clustered=true` matches only cluster representatives (which are a subset of representative instances).
active	`-priority=Suppressed -state=Invalid -state=Fixed -state=Later -finding="False Positive" -finding="Don't Care" similar=false clustered=true` This expression is interpreted as "find cluster representatives for warnings that do not have Priority equal to **Suppressed**, State equal to **Invalid**, **Fixed**, or **Later**,or Finding equal to **False Positive** or **Don't Care**". • On the left-hand side of an expression: `priority` refers to the Priority field, `state` refers to

	the State field, `priority` refers to the Finding field. • `Suppressed`, `Invalid`, `Fixed`, and `Later` don't need quote marks because they don't contain whitespace or special characters. `"False Positive"` and `"Don't Care"` do require quote marks, because they contain whitespace. • Every subexpression is negated with `-`. • `similar=false` matches only representative instances (all other instances have `similar=true`). • `clustered=true` matches only cluster representatives (which are a subset of representative instances). • In the absence of explicit ANDs or ORs, the expression is treated as if the subexpressions are joined by ANDs.

The following table shows simple example queries using `field-name`s from the warning search language.

Example	Explanation
`adesc="Version 1.0 of the thingy management system"`	Find warnings issued by analyses with exactly this description (case insensitively).
`adesc:stable`	Find warnings issued by analyses with descriptions containing the (case-insensitive) substring `stable`.

`adesc:none`	Find warnings issued by analyses with descriptions for which *either* of the following is true. • The description contains (case-insensitive) substring `none`. • The description contains no text, either because it has never been set or because it has been set and subsequently deleted. Note that the second of these is because `none` is a special value.
`aid=5`	Find warnings issued by the analysis with Analysis ID 5.
`aid=5 DIFFERENCE aid=6`	Find warning groups for which an instance was issued by the analysis with Analysis ID 5, but no instance was issued by the analysis with Analysis ID 6, then return a result set containing one instance (from analysis 5) of each group.
`(aid=5 DIFFERENCE aid=6) UNION (aid=6 DIFFERENCE aid=5)`	Find warning groups for which an instance was issued by one or the other of the two cited analyses but not both, then return a result set

	containing one instance (from analysis 5 or analysis 6) of each group.
`analysis="tms analysis 5"`	Find warnings issued by the analysis with this name (case insensitively).
`categories:CWE:123`	Find warnings with the (case-insensitive) substring `CWE:123` in their class categories.
`class:Overrun`	Find warnings with the (case-insensitive) substring `Overrun` in their class name.
`class=Leak`	Find warnings whose class is Leak (case insensitively).
`class=~~buffer%run`	Find warnings whose class name starts with 'buffer' and ends with 'run'. (case-insensitive)
`cluster=b18eab1ca9a4ee7d & aid=4 & similar:F`	Find the representative instances of all warnings that were issued in the analysis with Analysis ID 4 and belong to the warning cluster identified by hash `b18eab1ca9a4ee7d`.
`clustered:F & similar:T`	Find all instances of all warnings.
`clustered:F & similar:F`	Find the representative instances of all warnings.

`clustered:T`	Find the cluster representatives of all warning clusters (not affected by the setting of `similar`).
`detected="last 1 week"`	Find warnings detected between one week ago and now. One week ago is defined to be 7*24*60*60 seconds ago.
`directory=/u/src/mymodule`	Find warnings issued in a file in (case-insensitive) directory `/u/src/mymodule`. (See also the note above.)
`file=foo.c`	Find warnings issued in a file with basename `foo.c` (case insensitively). (See also the note above.)
`file=~"get.*\\.c"`	Find warnings issued in files whose basename contains substring 'get' and subsequently substring '.c' (case insensitive).
`fingerprint=4b655d73394c4a9b`	Find warnings whose Fingerprint is `4b655d73394c4a9b`.
`finding="True Positive"`	Find warnings whose Finding field is set to **True Positive**. Note: The string comparison is case-insensitive. However, since the Finding value is selected from a list, you will

	generally not have values that differ only in case. (This is also true for Priority and State).
`firstdetected=2008`	Find warning instances in warning groups that were first created in the year 2008.
`id=67.10254`	Find the warning with Warning ID 67.10254
`id=67.`	Find warnings with Group ID 67.
`id=.10254`	Find the warning with Instance ID 10254.
`id=67`	Find warnings whose Instance ID or Group ID is 67.
`iid=10254`	Find the warning with Instance ID 10254.
`iid=10254..10270`	Find warnings with Instance ID between 10254 and 10270 (inclusive).
`language=C`	Find warnings issued in a file with language C (not C++).
`line=42`	Find warnings issued on line 42 (of any file).
`line_content:SIGHUP`	Find warnings issued on a line whose code includes string `SIGHUP` (in any context).

`line_content_xml:"<s macro="1"><c>SIGHUP</c></s>"`	Find warnings issued on a line whose code includes macro `SIGHUP`.
`line_content_xml:"<s taint="`	Find warnings issued on a line whose code includes a tainted value.
`line=5 INTERSECT line=6`	Find all warning groups that have at least one instance issued on line 5 (of any file) **and** at least one instance issued on line 6 (of any file), then return a result set containing one instance (issued on either line 5 or line 6) of each group.
`listing:ThingyVisitor`	Find warnings with code listings that contain the (case-insensitive) substring `ThingyVisitor`.
`listing:emptyset`	Find warnings with code listings that contain the (case-insensitive) substring `emptyset`.
`listing:"emptyset("`	Find warnings with code listings that contain the (case-insensitive) substring `emptyset(`.
`listing_xml:main.c`	Find warnings with listing xml that contains the (case-insensitive) substring `foo.c`. This will include warnings with code listings containing code from a file whose name contains the

	(case-insensitive) substring `main.c`.
`modified=1/2..1/5`	Find warnings that were last modified (annotated, created, new similar warnings) between the beginning of 1/2 and the end of 1/5. 1/2 will be defined to be the last time it was 1/2, and 1/5 is defined to be the very next 1/5 after that 1/2. The interpretation of 1/2 and 1/5 depends on your Date Parse Formats setting. By default, 1/2 is interpreted as January 2nd and 1/5 as January 5th. If you would prefer that 1/2 be interpreted as February 1st and 1/5 as May 1st, change the settings in the Content tab of the Settings page.
`new_warning:true aid=5`	Find warnings that were first issued by the analysis with Analysis ID 5.
`notes:expensive`	Find warnings that have had notes added to them containing the (case-insensitive) substring `expensive`.
`owner=bob`	Find warnings belonging to bob. Note: The string comparison here is case-insensitive, but

	account usernames are (for internationalization reasons) case-sensitive. This search will find warnings owned by `bob`, `Bob`, `BoB`,....
`owner:none`	Find warnings for which *either* of the following is true. • The warning owner is a user whose user name contains (case-insensitive) substring `none`. • The warning has no owner: either because an owner has never been assigned, or because an owner has been assigned and subsequently removed. Note that the second of these is because `none` is a special value.
`path=/tmp/foo.c`	Find warnings issued in any file with the (case-insensitive) absolute name `/tmp/foo.c` (case insensitively).
`path:src/mod1/bar.c`	Find warnings issued in any file whose (case-insensitive) absolute

	name contains the substring `src/mod1/bar.c`.
`path="c:\\Program Files\\TMS\\foo.h"`	Find warnings issued in any file with the (case-insensitive) absolute name `c:\Program Files\TMS\foo.h`. Note that backslashes must be escaped. (See also the note above.)
`pdesc="thingy management system"`	Find warnings belonging to analyses of projects with this description (case-insensitively).
`pid=2`	Find warnings issued by analyses of the project with Project ID 2.
`priority:P0`	Find warnings whose priority contains the (case-insensitive) substring `P0`.
`priority:none`	Find warnings whose priority contains the (case-insensitive) substring `none`. Note that the other matching behaviors for special value `none` will never apply to `priority`: • It is not possible to have the empty string as a priority value. (It's not one of the built-in

	options, and the functionality for creating a new value requires a nonempty string.) • It is not possible to have no priority value at all.
`procedure:main`	Find warnings where the primary location is inside a procedure whose name contains the (case-insensitive) substring `main`.
`project=tms`	Find warnings belonging to analyses of the project named `tms` (case-insensitively).
`ptree_path:/TreeX/TreeA`	Find warnings belonging to analyses of: • projects that have an ancestor project tree whose PTree Name begins with string `TreeA`, and whose Parent Project Tree's PTree Name, *and* • projects whose Name begins with string `TreeA`, and whose Parent Project

	Tree's PTree Name is `TreeX`.
`rank:5.0..10.0`	Find warnings with rank between 5.0 and 10.0, inclusive.
`score:50..`	Find warnings with score greater than or equal to 49.5 (see note above).
`significance:security`	Find warnings whose warning class significance is "security".
`similar:F & owner:lindsay`	Find the representative instances of all warnings that belong to lindsay.
`sql="EXISTS (SELECT 5)"`	This Boolean condition will be inserted as a term in the WHERE clause used to implement the search. Only results satisfying this term will be returned.
`state=New`	Find warnings with State **New**. See the note on case-sensitivity for State.
`owner=bob & state=new`	Find warnings with state **New** that belong to bob.

	See the notes on case-sensitivity for State and case-sensitivity for Owner.
`(owner=bob | owner=george) & state=new`	Find warnings with state **New** that belong to bob or george. See the notes on case-sensitivity for State and case-sensitivity for Owner.
`-(owner=bob | owner=george) & state=new`	Find warnings with state **New** that do not belong to bob or george. See the notes on case-sensitivity for State and case-sensitivity for Owner.
`owner!=bob & state=new`	Find warnings with state **New** that do not belong to bob. See the notes on case-sensitivity for State and case-sensitivity for Owner.

SQL Terms

The `sql` field-name is provided to allow extra search customization, but is not supported. It is only available to users with `G SQL CONSOLE` permission.

For a search that includes a field-condition of the form

`sql:`*`cond`*

CodeSonar will obtain search results by executing an SQL statement of the form

```
SELECT <your specified columns>
   FROM <CodeSonar-determined relation>
```

```
WHERE <other-terms1 OP1> cond  <OP2 other-terms2>
ORDER BY <your specified order>
```

where

<your specified columns>	Depends on the set of columns you have chosen to display in the search result table.
<CodeSonar-determined relation>	Depends on *<your specified columns>* and *<other-terms$_{1,2}$>* but will be either `cs_warninginstance` or a join of `cs_warninginstance` and one or more other relations.
<other-terms$_{1,2}$>, *<OP$_{1,2}$>*	Depend on the other terms (specified or implicit) in your search, and the operators used to combine them.
cond	Is a Boolean expression suitable for evaluation in the WHERE clause of this query.
your specified order	Depends on the sort order(s) you have imposed on the search result table.

- In cases where the FROM clause does not include attributes that you want to test in the WHERE clause, you will need to include a suitable EXISTS statement in *cond*. For example, to find warnings whose Class includes "null", you could use

  ```
  sql:"EXISTS (SELECT * from cs_warningclass where
  lower(label_xml) LIKE '%null%' AND cs_warningclass.id
  = cs_warninginstdata.warningclass_id)"
  ```

  although in this case it is much simpler to use

  ```
  class:null
  ```

- To see samples of CodeSonar's SQL queries in general and WHERE clauses in particular, use your web browser to view the source of any warning search results page. The SQL query used to obtain the results shown on the page is included in an HTML comment directly above the result table.
- The hub database schema is available for inspection at `$CSONAR/codesonar/py/SCHEMA`.
  You can also use the SQL Console to examine the schema programmatically.