Goanna Studio User Guide

Version 3.6.4

Visual Studio & Atmel Studio Edition

November 23, 2015

Red Lizard Software
Australian Technology Park
Level 5, 13 Garden Street
Eveleigh NSW 2015
Australia

Web: http://www.redlizards.com
Support: support@redlizards.com

Contents

1 System Requirements

Before using Goanna, please check that your system and project meets the system requirements.

1.1 Operating Systems

Goanna supports the following versions of Windows:

- Windows XP (Service Pack 2 or higher)
- Windows Vista
- Windows 7
- Windows 8
- Windows 8.1
- Windows 10
- Windows Server 2003 (Service Pack 1 or higher)
- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2

Both 32-bit (x86) and 64-bit (x86-64/AMD64) versions of Windows are supported (except Windows XP and Windows Server 2003, which we only support 32-bit version).

Required Software

Before installing Goanna, you will need to install the following:

- Microsoft Visual C++ 2008 Redistributable (Download from http://www.microsoft.com/en-us/download/details.aspx?id=5582)
- .NET Framework 2.0 or higher (this is usually installed automatically when Visual Studio or Atmel Studio is installed)

1.1.1 Other Requirements

Some features of Goanna may require additional software or packages.

Requirements for Goanna Dashboard, HTML Report and Goanna Suppression Manager

Goanna Dashboard (see 6), HTML Report (see 3.3) and Goanna Suppression Manager (see 5.4) requires a web browser. The following web browsers are supported:

- Internet Explorer 9 or higher
- Microsoft Edge
- Mozilla Firefox - currently supported versions by Mozilla
- Google Chrome - currently supported versions by Google

While not officially supported, these features may also work on other modern browsers.

We also support Internet Explorer 7 and 8, however you may experience slow performance on these browsers; using Goanna Central with these browsers is not recommended.

Requirements for LM-X License Manager

Goanna uses LM-X License Manager 4.4.2 from X-Formation for licensing.

If you wish to use web-based UI of the License Manager (**Important:** Red Lizard Software does not provide full support for the web-based UI), the following software must be installed:

- A modern web browser
- Oracle Java Runtime Environment 1.6 or higher
- Adobe Flash Player

Please refer to X-Formation website (http://docs.x-formation.com/display/GEN/System+requirements+for+web-based+UIs) for more information.

1.2 IDE Requirements

1.2.1 Microsoft Visual Studio

Goanna Studio supports the following versions of Visual Studio:

- Visual Studio 2005
- Visual Studio 2008
- Visual Studio 2010
- Visual Studio 2012
- Visual Studio 2013
- Visual Studio 2015

Goanna Studio supports all editions of Visual Studio, except Express edition.

Important: Goanna Studio does not support any other versions of Visual Studio, including Visual Studio 6, Visual Studio .NET 2002 and Visual Studio .NET 2003.

There is a separate installation package for each version of Visual Studio. Please make sure to download the right package for your version of Visual Studio.

1.2.2 Atmel Studio

Goanna Studio supports the following versions of Atmel Studio:

- Atmel Studio 6.0
- Atmel Studio 6.1

Important: Goanna Studio does not support any other versions of Atmel Studio, including AVR Studio 5 and older. In addition, Goanna Studio does not currently support Atmel Studio 6.2 and Atmel Studio 7.0 Beta.

There is a separate installation package for each version of Atmel Studio. Please make sure to download the right package for your version of Atmel Studio.

1.3 Hardware Requirements

Goanna requires, at the minimum, the following hardware:

- Processor: Intel Pentium 4 or higher
- Memory: 1 GB or more
- Storage: Minimum 1 GB of free disk space

For optimal analysis performance, we recommend at least the following:

- Processor: Intel Core 2 Duo or later CPU with minimum speed 2 GHz. Multi core CPUs are recommended.
- Memory: 4 GB or more
- Storage: 5 GB or more of free disk space

For large projects, Goanna may require more RAM and disk space than the ones shown here.

1.4 Supported Compilers

Goanna currently supports the following compilers:

Compiler Name	Goanna Dialect Name	Common Compiler Executables
GNU (available on Atmel Studio only)	gnu	gcc, g++, ld
Microsoft Visual C++	`microsoft`	cl, link

1.4.1 A Word On C99 and C++11 Support

Goanna strives to support most C99 and C++11 features as long as the compilers used in your projects also accept them. However, please note that, for C++11 extensions, the analysis engine generally does not make use of these extensions or any information derived from usage of these extensions. For example, Goanna does not perform any pointer or memory use related analysis on C++11 `std::shared_ptr`.

1.4.2 A Word On Compiler-Specific Syntax Extensions

Goanna strives to support most compiler-specific C/C++ syntax extensions for supported compilers. However, please note that even in cases where Goanna supports compiler-specific C/C++ syntax extensions, the analysis engine will generally not make use of these extensions or any information derived from usage of these extensions.

1.5 Known Limitations

See 8.1 for a list of known limitations.

2 Getting Started

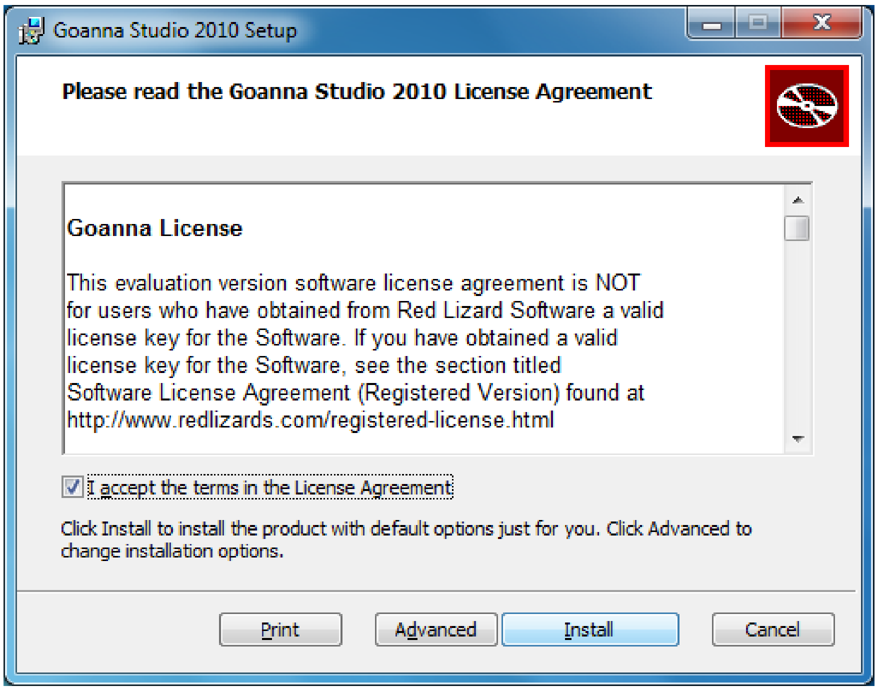
2.1 License Agreement

Before installing Goanna Studio, ensure you read the Goanna License Agreement, which can be found at
http://redlizards.com/license-term/evaluation-license-agreement/
if you are installing a trial version and
http://redlizards.com/license-term/
if you are installing a registered version of the tool. See Section 2.3 for how to purchase and install a license.

2.2 Installation

For the installation of Goanna Studio follow the next few steps:

1. Download the Goanna Studio distribution. The package is typically named `GoannaVS-X.msi`, where `X` is the release number.
2. Double Click the downloaded `.msi` file, and read and confirm the license agreement.



Goanna Studio 2010 Setup

Please read the Goanna Studio 2010 License Agreement

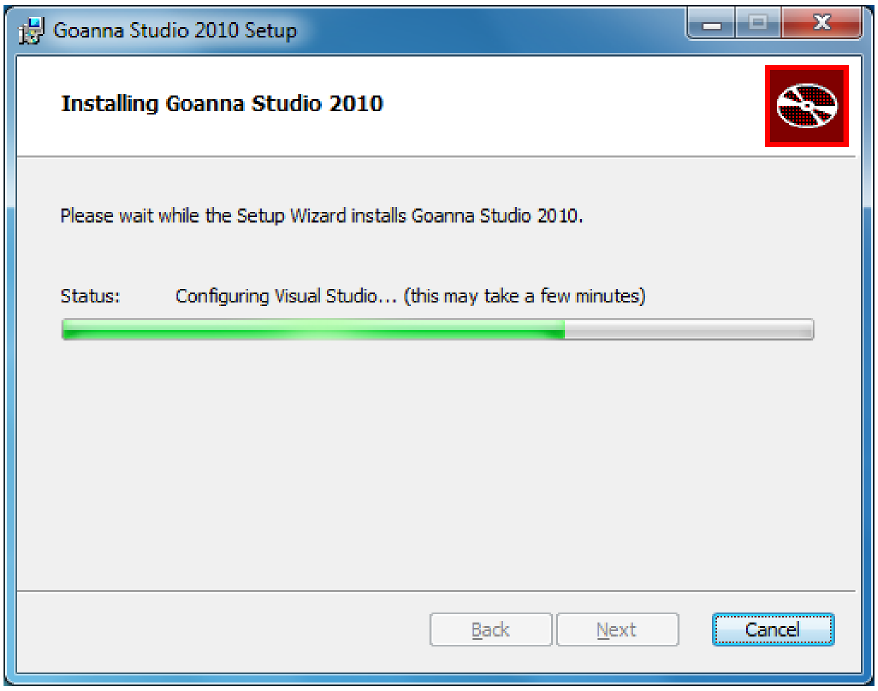
Goanna License

This evaluation version software license agreement is NOT for users who have obtained from Red Lizard Software a valid license key for the Software. If you have obtained a valid license key for the Software, see the section titled Software License Agreement (Registered Version) found at http://www.redlizards.com/registered-license.html

☑ I accept the terms in the License Agreement

Click Install to install the product with default options just for you. Click Advanced to change installation options.

3. Follow the installation wizard. Goanna Studio will invoke Visual Studio to complete the integration. This may take **several minutes**.

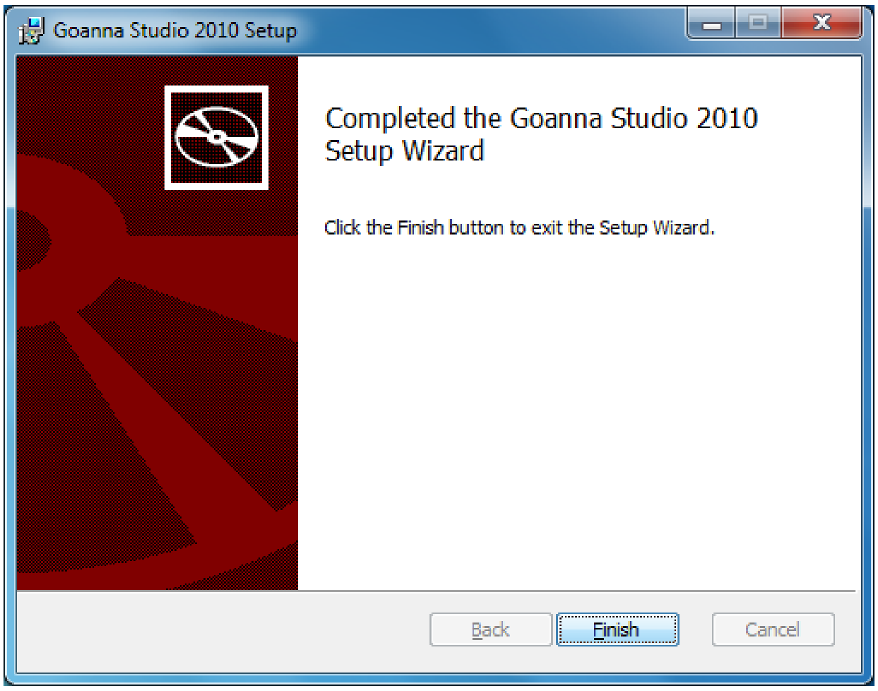

Goanna Studio 2010 Setup

Installing Goanna Studio 2010

Please wait while the Setup Wizard installs Goanna Studio 2010.

Status: Configuring Visual Studio... (this may take a few minutes)

4. At the end of the installation process click `Finish`.


Goanna Studio 2010 Setup

Completed the Goanna Studio 2010 Setup Wizard

Click the Finish button to exit the Setup Wizard.

5. Launch Visual Studio.

2.3 License Activation

Whether you are just evaluating Goanna or have purchased the full version, you must activate your license before you can use Goanna.

You should have received an email containing your license information from Red Lizard Software. This email will contain an Order Number that you will need below.

Then start the activation process by either:

- Simply opening any C/C++ solution in Visual Studio. At first run Goanna automatically shows you the activation screen.
- Alternatively, go to the Visual Studio Goanna menu and select Activate Goanna to open the activation screen.

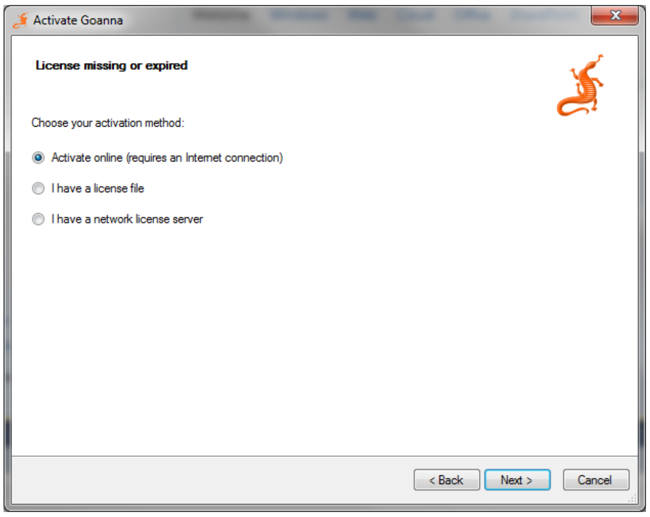

Activate Goanna
License missing or expired
Choose your activation method:
Activate online (requires an Internet connection)
I have a license file
I have a network license server
< Back
Next >
Cancel

Figure 1: Activation screen

2.3.1 Online Activation

At this point, you can now activate your license over the Internet connection. Follow these steps to complete the activation:

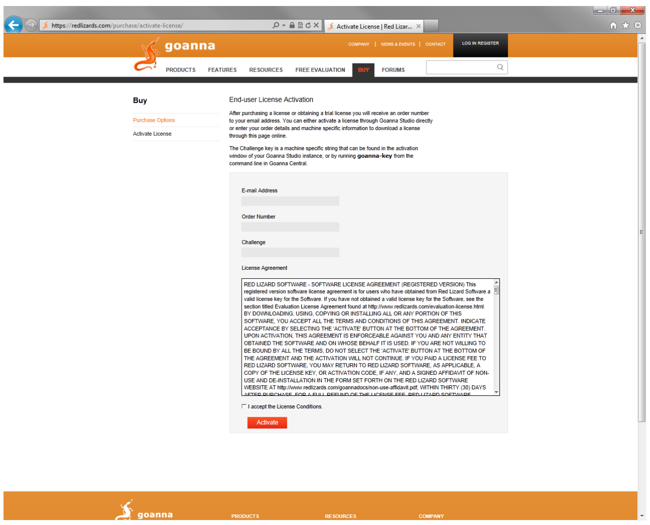
1. Select Activate online at the activation screen.
2. Read and agree the license agreement.
3. Type your e-mail address and the order number you received.
4. Press Next to finish activation.

Atmel Studio users: Atmel Gallery version of Goanna Studio comes with free 30-day evaluation license. To activate the evaluation license, select Activate my FREE 30-day evaluation and follow the instructions.

2.3.2 Manual Activation

Alternatively, you can also manually activate your license by obtaining license file at the Red Lizard Software website first. To do so, follow these steps:

1. Select I have a license file at the activation screen.
2. The screen will then show the challenge key of your machine.
3. Go to http://www.redlizards.com/purchase/activate-license/.
4. Type in your e-mail address and the order number.
5. Copy the challenge key into challenge text box.



End-user License Activation

After purchasing a license or obtaining a trial license you will receive an order number to your email address. You can either activate a license through Goanna Studio directly or enter your order details and machine specific information to download a license through this page online.

The Challenge key is a machine specific string that can be found in the activation window of your Goanna Studio instance, or by running **goanna-key** from the command line in Goanna Central.

RED LIZARD SOFTWARE - SOFTWARE LICENSE AGREEMENT (REGISTERED VERSION) This registered version software license agreement is for users who have obtained from Red Lizard Software a valid license key for the Software. If you have not obtained a valid license key for the Software, see the section titled Evaluation License Agreement found at http://www.redlizards.com/evaluation-license.html. BY DOWNLOADING, USING, COPYING OR INSTALLING ALL OR ANY PORTION OF THIS SOFTWARE, YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT. INDICATE ACCEPTANCE BY SELECTING THE 'ACTIVATE' BUTTON AT THE BOTTOM OF THE AGREEMENT. UPON ACTIVATION, THIS AGREEMENT IS ENFORCEABLE AGAINST YOU AND ANY ENTITY THAT OBTAINED THE SOFTWARE AND ON WHOSE BEHALF IT IS USED. IF YOU ARE NOT WILLING TO BE BOUND BY ALL THE TERMS, DO NOT SELECT THE 'ACTIVATE' BUTTON AT THE BOTTOM OF THE AGREEMENT AND THE ACTIVATION WILL NOT CONTINUE. IF YOU PAID A LICENSE FEE TO RED LIZARD SOFTWARE, YOU MAY RETURN TO RED LIZARD SOFTWARE, AS APPLICABLE, A COPY OF THE LICENSE KEY, OR ACTIVATION CODE, IF ANY, AND A SIGNED AFFIDAVIT OF NON-USE AND DE-INSTALLATION IN THE FORM SET FORTH ON THE RED LIZARD SOFTWARE WEBSITE AT http://www.redlizards.com/goannadocs/non-use-affidavit.pdf, WITHIN THIRTY (30) DAYS AFTER PURCHASE, FOR A FULL REFUND OF THE LICENSE FEE. RED LIZARD SOFTWARE

Figure 2: Activation web page

6. Read and agree the license agreement, then press Activate.
7. Download your license file via the generated download link, and choose this file in the activation screen.
8. Press Next in the activation screen to finish activation.

2.3.3 Activating Floating License

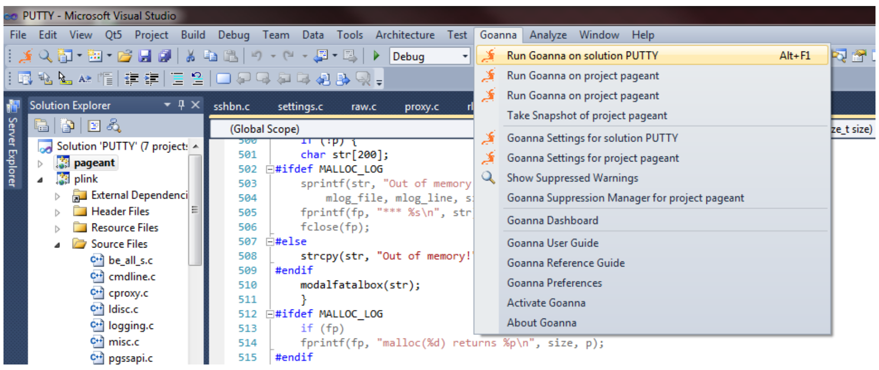
If you have floating (network) license, follow these steps to borrow a seat from the license server:

1. Select I have a network license server at the activation screen.
2. Type the host name, and optionally port number, of your license server.
3. Enter the borrow duration.
4. Press Next to finish activation.

3 Using Goanna Studio

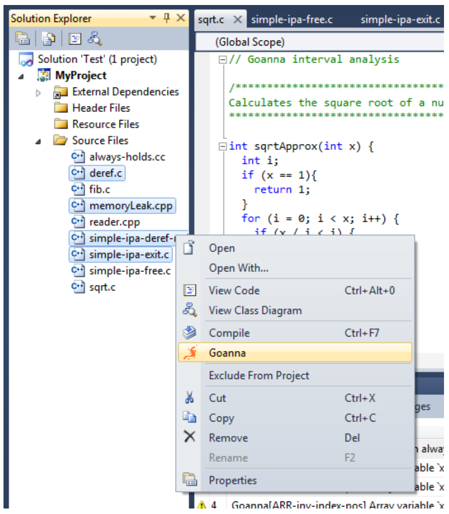
3.1 Running Goanna Studio on a Project

Quick Solution Select the files, project or solution you want analyzed in the Project Explorer and click the Goanna icon in the toolbar. Goanna Studio will automatically display warnings, if there are any. ☐


PUTTY - Microsoft Visual Studio
File Edit View Qt5 Project Build Debug Team Data Tools Architecture Test Goanna Analyze Window Help
Debug
Run Goanna on solution PUTTY Alt+F1
Run Goanna on project pageant
Run Goanna on project pageant
Take Snapshot of project pageant
Goanna Settings for solution PUTTY
Goanna Settings for project pageant
Show Suppressed Warnings
Goanna Suppression Manager for project pageant
Goanna Dashboard
Goanna User Guide
Goanna Reference Guide
Goanna Preferences
Activate Goanna
About Goanna
Server Explorer
Solution Explorer
Solution 'PUTTY' (7 project
pageant
plink
External Dependenci
Header Files
Resource Files
Source Files
be_all_s.c
cmdline.c
cproxy.c
ldisc.c
logging.c
misc.c
pgssapi.c
sshbn.c settings.c raw.c proxy.c
(Global Scope)
ze_t size)

```
    char str[200];
#ifdef MALLOC_LOG
    sprintf(str, "Out of memory
        mlog_file, mlog_line, s
    fprintf(fp, "*** %s\n", str
    fclose(fp);
#else
    strcpy(str, "Out of memory!
#endif
    modalfatalbox(str);
    }
#ifdef MALLOC_LOG
    if (fp)
    fprintf(fp, "malloc(%d) returns %p\n", size, p);
#endif
```

Figure 3: Analysis of an entire project.



```
// Goanna interval analysis

/*********************************
Calculates the square root of a nu
**********************************

int sqrtApprox(int x) {
  int i;
  if (x == 1){
    return 1;
  }
  for (i = 0; i < x; i++) {
```

```
4   Goanna[ARR-inv-index-pos] Array variable `x
```

Figure 4: Analysis of selected files.

Goanna Studio can be invoked on a project or a solution by selecting the appropriate entry from the Goanna menu, where the Goanna icon appears (see Fig. 3). You can also select several files and right-click on these. The Goanna option in the context menu allows you to run the tool over the selected files only (Fig. 4). Goanna can be called in the same way from the context menu of a project or a solution in the Project Explorer.

3.1.1 Warning Messages

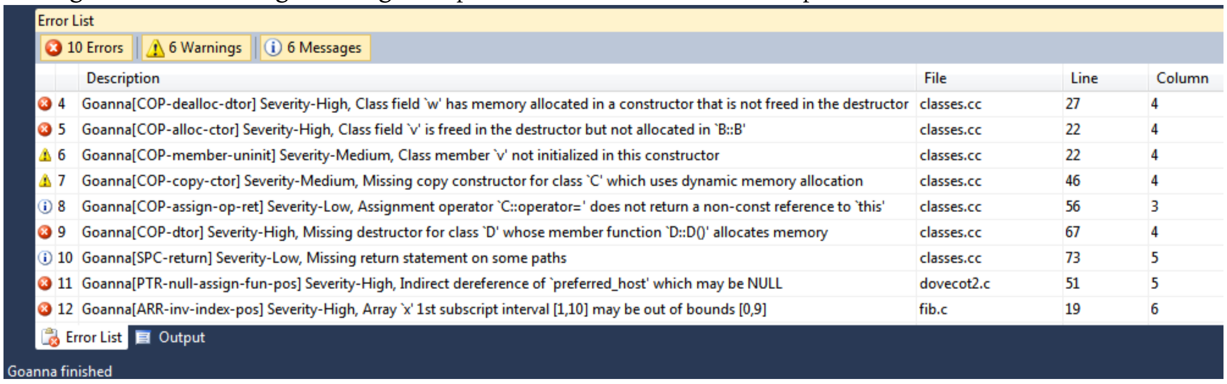
Any results from Goanna Studio will appear in the Error List window. The description of each warning takes the following format:

```
Goanna[SPC-return] Missing return statement on some paths. CERT-MSC37-C
    CWE-758 MISRAC2004-16.8 MISRAC++2008-8-4-3
```

The description consists of several parts:

- Goanna to distinguish Goanna warnings from other sources of markers in the Error List view.
- The *check name* (for example, `[SPC-return]`. This is the unique identifier of the Goanna check, used everywhere to identify the check, such as in the Reference Guide, the Describe Check help information, and in the Checks tab of Goanna's project/solution settings.
- The text of the warning message itself, for example, `Missing return statement on some paths`.
- A list of rule(s) from coding standards to which the warning applies, for example, `CERT-MSC37-C CWE-758 MISRAC2004-16.8 MISRAC++2008-8-4-3`. This cross references Goanna warnings back to the corresponding rule(s) in coding standards, whether or not you have enabled all the checks in those standards.

Each warning also identifies the file and line number, in much the same way as the compiler's warning and error messages. Double-clicking a message will place the cursor at the relevant part of the code.


Error List

⊗ 10 Errors | ⚠ 6 Warnings | ⓘ 6 Messages

		Description	File	Line	Column
⊗	4	Goanna[COP-dealloc-dtor] Severity-High, Class field `w' has memory allocated in a constructor that is not freed in the destructor	classes.cc	27	4
⊗	5	Goanna[COP-alloc-ctor] Severity-High, Class field `v' is freed in the destructor but not allocated in `B::B'	classes.cc	22	4
⚠	6	Goanna[COP-member-uninit] Severity-Medium, Class member `v' not initialized in this constructor	classes.cc	22	4
⚠	7	Goanna[COP-copy-ctor] Severity-Medium, Missing copy constructor for class `C' which uses dynamic memory allocation	classes.cc	46	4
ⓘ	8	Goanna[COP-assign-op-ret] Severity-Low, Assignment operator `C::operator=' does not return a non-const reference to `this'	classes.cc	56	3
⊗	9	Goanna[COP-dtor] Severity-High, Missing destructor for class `D' whose member function `D::D()' allocates memory	classes.cc	67	4
ⓘ	10	Goanna[SPC-return] Severity-Low, Missing return statement on some paths	classes.cc	73	5
⊗	11	Goanna[PTR-null-assign-fun-pos] Severity-High, Indirect dereference of `preferred_host' which may be NULL	dovecot2.c	51	5
⊗	12	Goanna[ARR-inv-index-pos] Severity-High, Array `x' 1st subscript interval [1,10] may be out of bounds [0,9]	fib.c	19	6

3.1.2 Parse Errors

Goanna identifies syntactically erroneous code, ensuring that analysis only takes place on syntactically correct code. If a file contains syntax errors Goanna will simply skip analysis of that file, and report the parse errors in the error list.

3.1.3 Check Descriptions

The Goanna Checks tab allow you to view an online version of the check descriptions that appear in the companion reference manual *Goanna Reference Manual*. Simply right-click a warning message and select Describe Check from the context menu to view a detailed description of the check, including simplified code examples.

3.1.4 Macro Visualisation

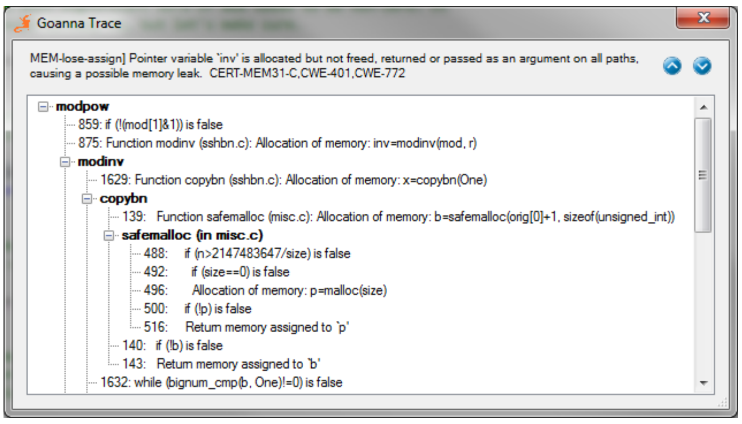
Sometimes it is difficult to determine the reason for a particular warning, because the code invokes a C preprocessor (C++) macro. Visual Studio offers a tooltip window showing the macro definition, but that may not be enough for effective debugging.

The Visual Studio edition of Goanna Studio can show the expansion of C++ macros. Right-click anywhere in a Visual Studio code window to get a context menu, and choose Show macros. Each C++ macro in the source file will be highlighted with a blue underline. Hover over the macro to show its expansion in a tooltip window.

While macros are being displayed, any purple underlines indicating Goanna warnings are hidden. From the same context menu you used to show the macros, choose Clear macros to remove the blue macro underlines and restore the purple warning underlines.

3.1.5 Traces

The Goanna trace window will allow you to trace through the path of execution leading to a warning message. This may help with the debugging of larger functions. Simply right-click the warning message and select Show trace from the context menu to view the Goanna Trace window.



MEM-lose-assign] Pointer variable 'inv' is allocated but not freed, returned or passed as an argument on all paths, causing a possible memory leak. CERT-MEM31-C,CWE-401,CWE-772

```
⊟ modpow
  ├ 859: if (!(mod[1]&1)) is false
  ├ 875: Function modinv (sshbn.c): Allocation of memory: inv=modinv(mod, r)
  ⊟ modinv
    ├ 1629: Function copybn (sshbn.c): Allocation of memory: x=copybn(One)
    ⊟ copybn
      ├ 139:   Function safemalloc (misc.c): Allocation of memory: b=safemalloc(orig[0]+1, sizeof(unsigned_int))
      ⊟ safemalloc (in misc.c)
        ├ 488:    if (n>2147483647/size) is false
        ├ 492:      if (size==0) is false
        ├ 496:      Allocation of memory: p=malloc(size)
        ├ 500:    if (!p) is false
        └ 516:    Return memory assigned to 'p'
      ├ 140:  if (!b) is false
      └ 143:  Return memory assigned to 'b'
    ├ 1632: while (bignum_cmp(b, One)!=0) is false
```

Figure 5: Trace window

3.1.6 Stopping Goanna Analysis

To stop Goanna analysis that is currently running, open Goanna menu and click Cancel Goanna.

Note: This terminates **all** Goanna analysis that are currently running on your machine, not just the one that is running within Visual Studio or Atmel Studio IDE.

3.2 Goanna Solution/Project Settings

Goanna provides various configuration options which apply to either whole solution, or a project. You can access the settings window in two ways:

- Open Goanna menu and click either:
  - Goanna Settings for solution <*name of solution*> or
  - Goanna Settings for project <*name of project*>

  or
- Right-click a solution or a project in the Solution Explorer, and click Goanna Solution Settings or Goanna Project Settings.

Solution settings apply to all projects within that solution. If you wish to apply different settings to some projects, in the Project Settings window uncheck Inherit settings from Solution.

Goanna settings for a project or solution are divided into a number of tabs: Checks, Files, and Options.

3.2.1 Available Goanna Checks

The Checks tab lets you enable and disable the checks that Goanna performs.

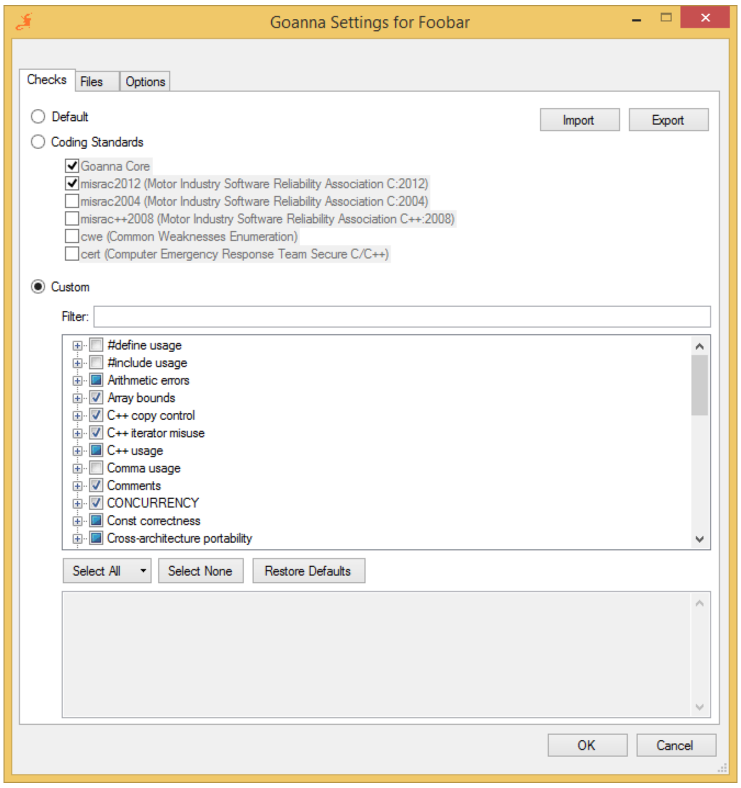
To configure and select the checks that Goanna Studio performs in each run, you can select one of three options:

- Default: This selects the Goanna Core set of checks, which produce the most accurate and complete diagnostics for most projects. This option is recommended for most users.
- Coding Standards: This allows you, in addition to Goanna Core, to optionally include checks from a number of industry coding standards. This option is useful for checking compliance of your code with these standards. In addition to the default set of checks (Goanna Core), Goanna also supports the following standards:

Standard code	Standard name
`cert`	Computer Emergency Response Team (CERT) C/C++ Coding Standard
`cwe`	Common Weakness Enumeration (CWE)
`misrac2004`	Motor Industry Software Reliability Association (MISRA) C:2004
`misrac++2008`	Motor Industry Software Reliability Association (MISRA) C++:2008
`misrac2012`	Motor Industry Software Reliability Association (MISRA) C:2012

- Custom: This allows you enable or disable each check individually. Click on the checkboxes to toggle each check or group of checks. You can also use the Select All and Select None buttons when the Custom option is selected.

For a description of a specific check simply click on the check code and short description next to the checkbox. Details and examples of the checks can be found in the companion reference manual *Goanna Reference Manual*.


Goanna Settings for Foobar

Checks | Files | Options

○ Default
○ Coding Standards
- ☑ Goanna Core
- ☑ misrac2012 (Motor Industry Software Reliability Association C:2012)
- ☐ misrac2004 (Motor Industry Software Reliability Association C:2004)
- ☐ misrac++2008 (Motor Industry Software Reliability Association C++:2008)
- ☐ cwe (Common Weaknesses Enumeration)
- ☐ cert (Computer Emergency Response Team Secure C/C++)

● Custom

Filter:

- ☐ #define usage
- ☐ #include usage
- ■ Arithmetic errors
- ☑ Array bounds
- ☑ C++ copy control
- ☑ C++ iterator misuse
- ■ C++ usage
- ☐ Comma usage
- ☑ Comments
- ☑ CONCURRENCY
- ■ Const correctness
- ■ Cross-architecture portability

Figure 6: Goanna Settings for Checks

Checks can be filtered by using the Filter option, simply type in what you would like to filter by and the checks will be updated. Deleting any text in the Filter text box will display all the checks. Note that the Select All and Select None buttons apply to the filtered checks, easily allowing you to add remove checks by filtering.

For example, in Custom mode you can easily enable all the Common Weakness Enumeration (CWE) checks by typing cwe in the Filter box, and clicking Select All. Then clear the Filter box again to see the effect.

Exporting And Importing Check Selection

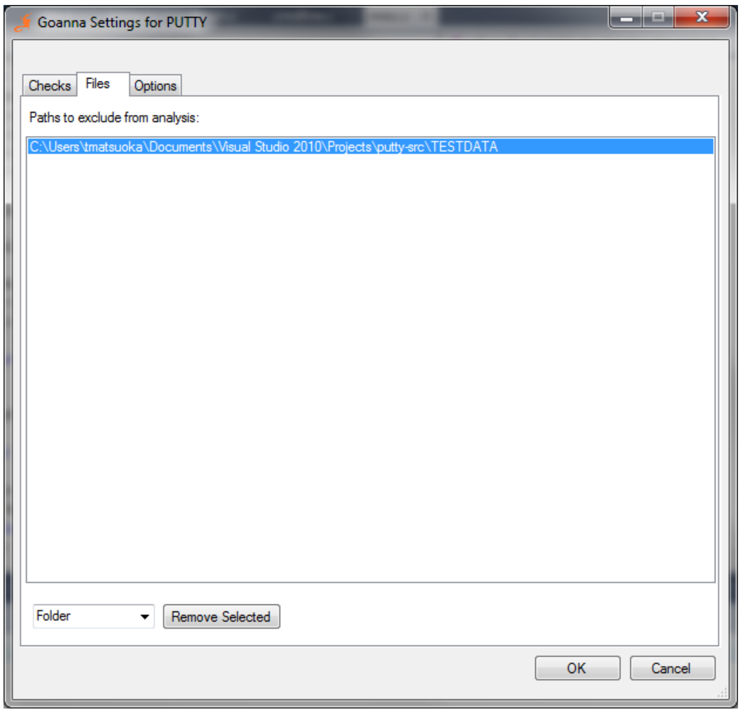
Goanna Studio also supports exporting and importing the check selection status so that you can easily configure multiple machines to use the same set of checks.

To export checks selection to a file, click Export button and specify the location of the file to be saved. Once the selection is exported to a file, you can then import this setting by clicking Import button.

Note that, the exported checks selection also contains information about what checks packages are enabled, to ensure all necessary checks are available. This means that, for example, if you copy checks selection to a different machine with different checks packages enabled, then this may result in enabling more checks than intended. Goanna Studio will warn at import time in such cases.

3.2.2 Files to Analyze

The Files tab lets you quickly include and exclude the files that Goanna analyzes.



Paths to exclude from analysis:

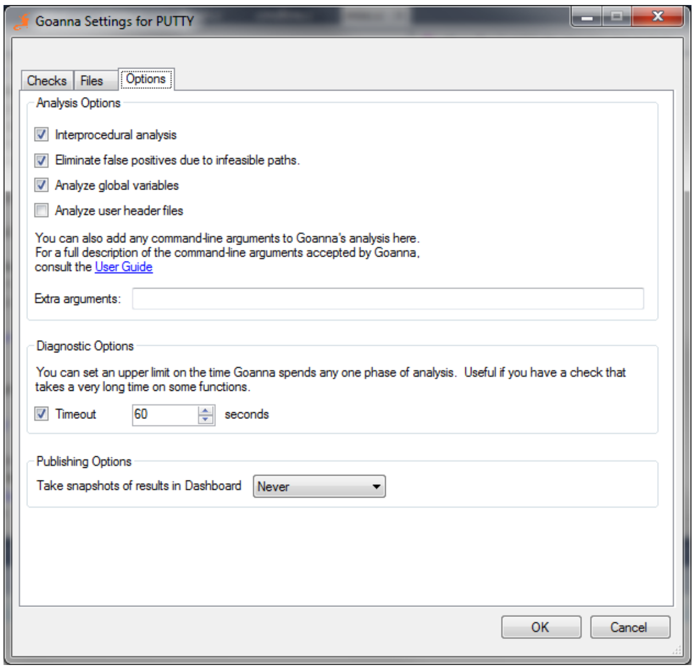
C:\Users\tmatsuoka\Documents\Visual Studio 2010\Projects\putty-src\TESTDATA

Figure 7: Files tab

- Paths to exclude from analysis: This is a list of files, paths and regular expressions that Goanna should not include in analysis. The drop down menu to the right allows the addition of new files, paths, or regular expressions. The Remove Selected button removes all selected items from the ignored paths list.

3.2.3 Options

The Options tab contains advanced options that control how Goanna analyzes your code.


Goanna Settings for PUTTY

Checks | Files | Options

Analysis Options

- [✓] Interprocedural analysis
- [✓] Eliminate false positives due to infeasible paths.
- [✓] Analyze global variables
- [ ] Analyze user header files

You can also add any command-line arguments to Goanna's analysis here.
For a full description of the command-line arguments accepted by Goanna,
consult the User Guide

Extra arguments:

Diagnostic Options

You can set an upper limit on the time Goanna spends any one phase of analysis. Useful if you have a check that takes a very long time on some functions.

[✓] Timeout 60 seconds

Publishing Options

Take snapshots of results in Dashboard: Never

OK | Cancel

Figure 8: Goanna Options

A description of these Goanna Studio options is as follows:

- Interprocedural analysis (default: yes): This enables interprocedural analysis. See section 4.1 later in this document for a detailed description of this feature.
- Eliminate false positives due to infeasible paths (default: yes): This enables false positive elimination based on a Satisfiability Modulo Theories (SMT) solver algorithm. See section 4.2 later in this document for a detailed description of this feature.
- Analyze global variables (default: yes): Instructs Goanna to include global variables in its analysis.
- Analyze user header files (default: no): Instructs Goanna to include user headers (generally, those included using `#include "..."` syntax) into analysis.
- Extra arguments (default: none): This allows you to pass arguments to the Goanna executable. Most users will not need to use this field, because Goanna Studio synthesizes the correct arguments from project files.

  In some cases, though, a user may wish to pass arguments not found in a project file. When running Goanna, you may want to search for include files in particular directories using the -I flag: `-I C:\SomeIncludePath`. A list of the available arguments to the Goanna executable is detailed in section 7.1.
- Timeout (default: 240 seconds): This is the maximum number of seconds Goanna is allowed to spend in each analysis phase within one source file.

  Generally speaking, increasing timeout may result in more accurate results, but will take longer to complete the analysis. Decreasing timeout will improve the running time, but may result in less accurate results. Due to the underlying technology of the Goanna analysis engine, this timeout is essential.

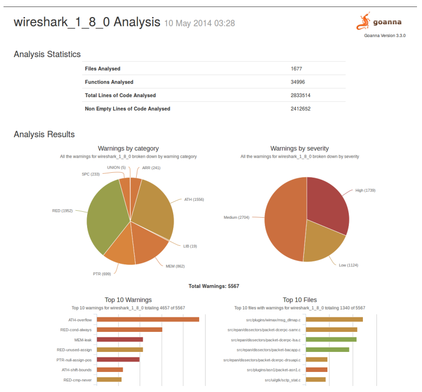
  Important: Setting this value to 0 (meaning infinite) is discouraged; this may cause Goanna to not terminate!
- Automatically add snapshot at each Goanna run (default: Always): By default Goanna automatically generates a new snapshot for the Dashboard (see 6) after each analysis run. You can change this to Never to never generate a snapshot automatically.

3.3 HTML Report of Analysis Results (Advanced)

Goanna can generate an HTML report of the analysis results. There are two types of HTML reports:

- *Summary Report*, which shows a summary of the analysis result, including:
  - Basic statistics, such as the number of files analyzed,
  - Per-category and per-severity warnings pie chart,
  - Bar charts of the top 10 warnings per check and top 10 files with warnings, and
  - List of warning numbers per check.

  This report shows the high-level statistics of the analysis results in an easy-to-see format; and
- *Warnings Report*, which shows a table of all warnings found; this table is interactive and supports filtering.



wireshark_1_8_0 Analysis 10 May 2014 03:28

Goanna Version 3.3.0

Analysis Statistics

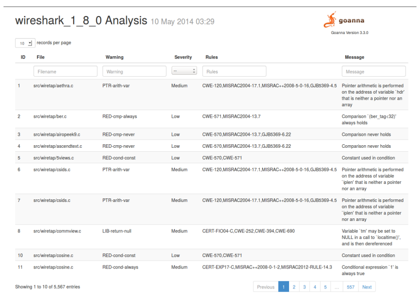
Files Analysed	1677
Functions Analysed	34996
Total Lines of Code Analysed	2833514
Non Empty Lines of Code Analysed	2412652

Analysis Results

Total Warnings: 5567

Warnings by category
All the warnings for wireshark_1_8_0 broken down by warning category
UNION (5)
ARR (241)
SPC (233)
ATH (1556)
RED (1952)
LIB (19)
MEM (862)
PTR (699)
Warnings by severity
All the warnings for wireshark_1_8_0 broken down by severity
High (1739)
Medium (2704)
Low (1124)
Top 10 Warnings
Top 10 warnings for wireshark_1_8_0 totaling 4657 of 5567
ATH-overflow
RED-cond-always
MEM-leak
RED-unused-assign
PTR-null-assign-pos
ATH-shift-bounds
RED-cmp-never
Top 10 Files
Top 10 files with warnings for wireshark_1_8_0 totaling 1340 of 5567
src/plugins/wimax/msg_dlmap.c
src/epan/dissectors/packet-dcerpc-samr.c
src/epan/dissectors/packet-dcerpc-lsa.c
src/epan/dissectors/packet-bacapp.c
src/epan/dissectors/packet-dcerpc-drsuapi.c
src/plugins/asn1/packet-asn1.c
src/ui/gtk/sctp_stat.c

Example of Summary Report



wireshark_1_8_0 Analysis 10 May 2014 03:29

Goanna Version 3.3.0

10 records per page

ID	File	Warning	Severity	Rules	Message
	Filename	Warning	--	Rules	Message
1	src/wiretap/aethra.c	PTR-arith-var	Medium	CWE-120,MISRAC2004-17.1,MISRAC++2008-5-0-16,GJB5369-4.5	Pointer arithmetic is performed on the address of variable `hdr' that is neither a pointer nor an array
2	src/wiretap/ber.c	RED-cmp-always	Low	CWE-571,MISRAC2004-13.7	Comparison `(ber_tag<32)' always holds
3	src/wiretap/airopeek9.c	RED-cmp-never	Low	CWE-570,MISRAC2004-13.7,GJB5369-6.22	Comparison never holds
4	src/wiretap/ascendtext.c	RED-cmp-never	Low	CWE-570,MISRAC2004-13.7,GJB5369-6.22	Comparison never holds
5	src/wiretap/5views.c	RED-cond-const	Low	CWE-570,CWE-571	Constant used in condition
6	src/wiretap/csids.c	PTR-arith-var	Medium	CWE-120,MISRAC2004-17.1,MISRAC++2008-5-0-16,GJB5369-4.5	Pointer arithmetic is performed on the address of variable `iplen' that is neither a pointer nor an array
7	src/wiretap/csids.c	PTR-arith-var	Medium	CWE-120,MISRAC2004-17.1,MISRAC++2008-5-0-16,GJB5369-4.5	Pointer arithmetic is performed on the address of variable `iplen' that is neither a pointer nor an array
8	src/wiretap/commview.c	LIB-return-null	Medium	CERT-FIO04-C,CWE-252,CWE-394,CWE-690	Variable `tm' may be set to NULL in a call to `localtime()', and is then dereferenced
10	src/wiretap/cosine.c	RED-cond-const	Low	CWE-570,CWE-571	Constant used in condition
11	src/wiretap/cosine.c	RED-cond-always	Medium	CERT-EXP17-C,MISRAC++2008-0-1-2,MISRAC2012-RULE-14.3	Conditional expression `1' is always true

Showing 1 to 10 of 5,567 entries

Example of Warnings Report

To generate HTML report files at the end of analysis, add **`--html-report`** option to Extra arguments field in the Options tab of the Goanna Settings dialog. This option accepts an optional argument to specify the type(s) of report to be generated:

- **`--html-report=summary`**: Generates summary report file.
- **`--html-report=warnings`**: Generates warnings report file.
- **`--html-report=all` or `--html-report`** with no extra argument: Generates both summary and warnings report files.
- **No `--html-report` option**: HTML report files will not be generated.

Generated HTML report files will be saved to the directory where your project(s) are located. (**Note:** Not the solution directory!)

3.4 Analysis Results In XML File (Advanced)

Goanna can optionally output the analysis result to a XML file. This is useful if you need Goanna to be used in conjunction with some other platform or framework and need programmatic access to the analysis result.

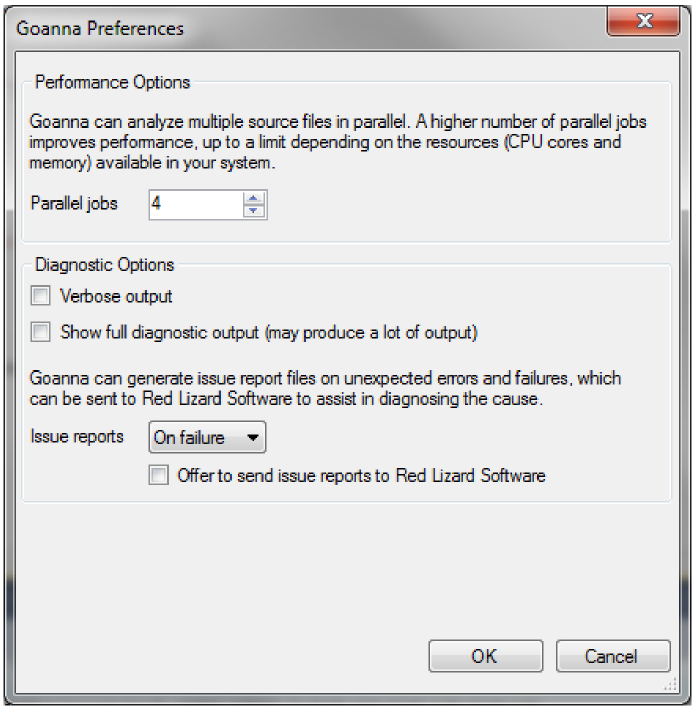
To generate a XML output file at the end of analysis, add **`--output-xml=<xml-file-name>`** option to Extra arguments field in the Options tab of the Goanna Settings dialog.

Generated XML output file will be saved to the specified file name in the directory where your project(s) are located. (**Note:** Not the solution directory!)

Note: If the specified XML output file already exists, Goanna will overwrite the file at the end of next analysis.

3.5 Goanna Global Preferences

In addition to per-solution and per-project settings, Goanna also has some global settings options. To access these options, click Goanna Preferences in Goanna menu.



Goanna Preferences

Performance Options

Goanna can analyze multiple source files in parallel. A higher number of parallel jobs improves performance, up to a limit depending on the resources (CPU cores and memory) available in your system.

Parallel jobs: 4

Diagnostic Options

☐ Verbose output

☐ Show full diagnostic output (may produce a lot of output)

Goanna can generate issue report files on unexpected errors and failures, which can be sent to Red Lizard Software to assist in diagnosing the cause.

Issue reports: On failure

☐ Offer to send issue reports to Red Lizard Software

Figure 10: Goanna Global Preferences Window

A description of these Goanna Studio options is as follows:

- Parallel Jobs: This option specifies how many (maximum) Goanna instances should run at one time (from 1 core up to 20 cores). By default this is set to the number of CPU cores available on the system.
- Verbose output: Enables verbose output mode, which causes Goanna to display command line arguments and Goanna output from the console.
- Show full diagnostics output: Enables even more verbose output, which includes full Goanna debugging output.
  Important: You should not enable this option unless you experience Goanna failures or crashes. Enabling this option will also slow down analysis and produce a lot of output.
- Issue reports: Goanna can automatically generate issue reports when you experience Goanna failures or errors, which can then be sent to Red Lizard Software to assist diagnosis. You can choose when to generate issue reports from the following options:
  - Never: Never generate issue reports.
  - On failure (default for non-evaluation users): Generate issue reports when Goanna experiences fatal error.
  - On error (default for evaluation users): Generate issue reports when Goanna experiences fatal error or PARSE error.
  - On timeout: Generate issue reports when Goanna experiences fatal error, PARSE error, or when analysis times out.
  - Always **(Not Recommended)**: Always generate issue reports after every analysis, *even if successful*.
- Offer to send issue reports to Red Lizard Software: If this option is checked, then when Goanna generates issue reports they will be displayed in a window which allows you to send the issue reports to Red Lizard Software for diagnosis.
  Important Note: Issue reports contain your name or e-mail address (extracted from license file), and additionally some information extracted from source files themselves. Please examine the content of the report carefully before submitting them to Red Lizard Software.

3.6 Checks Packages

Goanna Studio comes with a default package of checks that you can choose to enable via the Checks tab as described in Section 3.2.1. It is also possible to install more checks, specifically for the purpose of providing extra conformance checking for selected C/C++ standards.

Goanna ships with the following checks packages:

- **`stdchecks`** (enabled by default): "Goanna Core" checks. This checks package contains a set of checks for common C/C++ issues.
- **`security`**: Checks that are targetting well known security vulnerabilities.
- **`misrac2004`**: Dedicated checks for MISRA-C:2004 coding standard.
- **`misrac++2008`**: Dedicated checks for MISRA C++:2008 coding standard.
- **`misrac2012`**: Dedicated checks for MISRA C:2012 coding standard.

It is also possible to add custom checks packages to the analysis results. Contact Red Lizard Software for more information about custom checks packages.

To access these options, click Goanna Preferences in Goanna menu and select the Checks Packages tab.

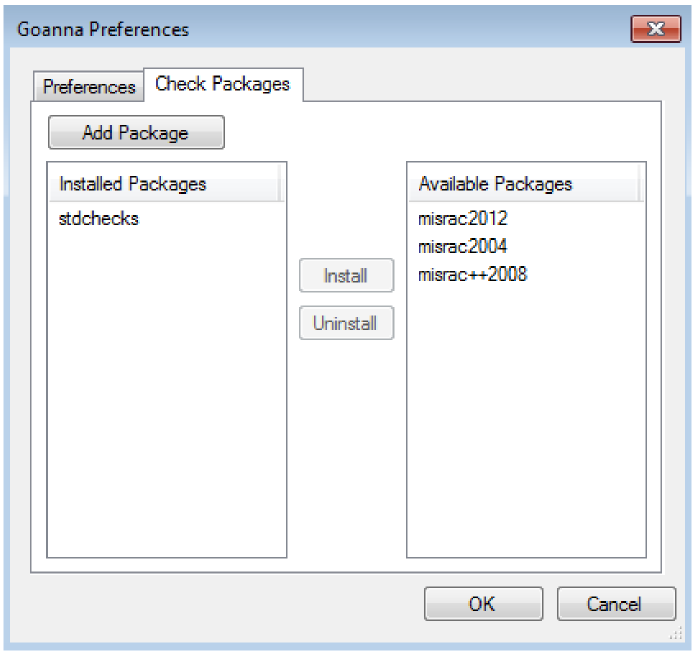


Figure 11: Goanna Checks Package Preferences

To enable a checks package, click the checks package you wish to enable from Available Packages list, then click Install. To disable a checks package, click the checks package you wish to disable from Installed Packages list, then click Uninstall.

If you have a custom checks package, then you can add it by clicking Add Package.

Important Note

Checks package operations are global and may affect analysis on all existing *and new* projects. Additionally, enabling a checks package will also change the default set of checks to be enabled for all existing *and new* projects.

Installing or uninstalling checks packages, and adding custom checks packages may require User Account Control (UAC) approval on Windows Vista, Windows Server 2008 and later versions of Windows, as these operations require write access to the installation directory of Goanna Studio.

3.7 Sample Code

Important: This sample code package can only be used with Visual Studio; it is not compatible with Atmel Studio.

A package containing a number of sample C/C++ files is available on our website. Go to http://www.redlizards.com/resources/example-code/ and download the `Visual Studio Sample Code` package corresponding to your version of Visual Studio. The files in this package may be useful for practicing using Goanna Studio, or ensuring that Goanna is working correctly.

A project file and solution file for Visual Studio are included in this directory. Open either of these files to open the solution.

Once the project has loaded, you can try analyzing it with Goanna Studio.

4 Getting the Best Results from Goanna

4.1 Interprocedural Analysis

Goanna's interprocedural analysis propagates information about function behaviour to other functions. This information includes parameter values, return values, and function effects that may impact other parts of the code. This enables Goanna to detect things in your program such as freeing of memory through function calls, functions that never return, and input values to some functions.

Interprocedural analysis is not limited to a specific set of checks, but rather enhances the precision of many checks.

An example of what interprocedural analysis can find can be seen in the sample of function `myAlloc`.

```
void *myAlloc(int param){
  void *p = malloc(param);
  if (p)
     return p;
   else
     return NULL;
}

int main(int argc, char ** argv) {
  int * n;
  n = (int*)myAlloc(sizeof(int) * 10);
  n[0] = 5; // this may be a dereference of NULL
  return *n;
}
```

Here, Goanna learns that `myAlloc` may return `NULL`. This means that when the return value of `myAlloc` is assigned to `n`, Goanna knows this value may be `NULL`. Therefore, the expression `n` may be dereferencing a `NULL` pointer, and Goanna will warn accordingly.

There is some additional computation overhead in running interprocedural analysis. If you need rapid results without much depth, then turning off interprocedural analysis will provide faster results (at the cost of accuracy in some checks). To turn off interprocedural analysis, use the `--no-ipa` option.

By default interprocedural analysis does two passes (in optimized order) over each file. This provides a good approximation for function behaviours, but may miss some complex behaviours that require many passes to accurately detect. Additional precision can be gained by increasing the iteration limit (the maximum number of passes Goanna will do). To change the interprocedural analysis iteration limit, use the `--ipa-iterations` option.

4.2 A Word on False Positives

Goanna considers all possible execution paths in your program, and will warn you if it finds potential defects (such as use of an uninitialised variable) that occur only on particular execution paths and not others. But sometimes, the execution path leading to a potential defect is actually not possible when the program is executed. If Goanna is able to deduce this through static analysis, then it won't warn you. But if it can't, then you may receive a spurious warning for a defect that isn't really there. Such warnings are called *false positives*.

Some false positives occur because Goanna currently does not track dependencies between variables in loops. For example, if you have a loop with two counters and only test one:

```
char buffer[11];
int i, count;

i = 0;
count = 10;
while (--count >= 0) {
  buffer[i++] = 'x';
}
buffer[i] = '\0';
```

Goanna may issue a false positive warning because it doesn't deduce that `i = 10` when the loop terminates.

Such false positives can often be suppressed with the `assert` macro (sec. 4.4). Otherwise you can suppress false positives using the Goanna Dashboard (see 6.2).

4.3 Using the _GOANNA Preprocessor Symbol

Goanna has a built-in preprocessor definition, defined by the macro

```
#define _GOANNA 1
```

This allows code to be explicitly included in or excluded from analysis by Goanna. For example:

```
#ifdef _GOANNA
//Code only to be included while the program is being analysed
#endif
#ifndef _GOANNA
//Code not to be analyzed by Goanna
#endif
```

4.4 Using the `assert` macro

Goanna can sometimes use information provided by `assert()` to refine its analysis of numerical and pointer values. It does this by using `assert` statements as assumptions for value ranges and pointer validity.

For example, in the code below:

```
  void my_fun(void) {
    int my_array[20];
    int x = rand();

    assert(x == 10);
    f(my_array[x]);
  }
```

the `assert()` means that the array reference must be in-bounds, even though the index variable x has a randomly-assigned value. Therefore, Goanna does not issue an out-of-bounds warning.

5 Suppressing Warnings

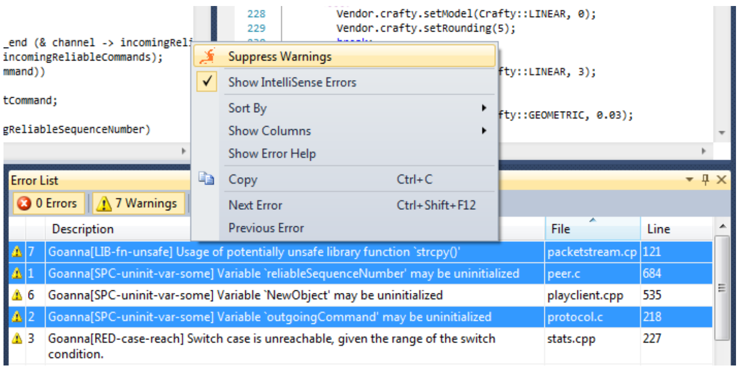
For a number of reasons, you may wish to ignore specific warning messages without modifying your check settings. Goanna Studio allows you to easily hide and show warnings to help keep emphasis on the most critical bugs.

5.1 Suppressing Warnings

Quick Solution Right-click a warning and select Suppress Warning from the context menu. To view all suppressed warnings, click the Show Suppressed Warnings button 🔍 in the toolbar. From there, you can unsuppress warnings in the same way. □

To hide a warning, right-click it in the Goanna Warnings list, and select Suppress Warning from the context menu.

Goanna takes special care to ensure that warning suppression never silently conceals actual regressions (bugs that are fixed but inadvertently reintroduced) or unrelated bugs. If you suppress a warning, and later modify the function or any other code having an effect on it, Goanna will notice that the code has been edited, and re-issue the warning. Internally Goanna uses a hash of the actual syntax to determine if the code has changed, so only significant changes will cause the warning to reappear, not just modifying the file or changing whitespace or comments.


```
_end (& channel -> incomingReli
incomingReliableCommands);
mmand))

tCommand;

gReliableSequenceNumber)
```

```
            Vendor.crafty.setModel(Crafty::LINEAR, 0);
            Vendor.crafty.setRounding(5);
fty::LINEAR, 3);
fty::GEOMETRIC, 0.03);
```

		Description	File	Line
⚠	7	Goanna[LIB-fn-unsafe] Usage of potentially unsafe library function `strcpy()'	packetstream.cp	121
⚠	1	Goanna[SPC-uninit-var-some] Variable `reliableSequenceNumber' may be uninitialized	peer.c	684
⚠	6	Goanna[SPC-uninit-var-some] Variable `NewObject' may be uninitialized	playclient.cpp	535
⚠	2	Goanna[SPC-uninit-var-some] Variable `outgoingCommand' may be uninitialized	protocol.c	218
⚠	3	Goanna[RED-case-reach] Switch case is unreachable, given the range of the switch condition.	stats.cpp	227

5.2 Showing Suppressed Warnings

When you ask Goanna Studio to ignore a warning, it will not be removed permanently from the database. The list of Goanna warnings in the Goanna Warnings tab can be toggled, to show either the suppressed or unsuppressed warnings. Click the Show Suppressed Warnings button 🔍 on the right-hand side of the Goanna Warnings tab to toggle which warnings are shown.

5.3 Unsuppressing Warnings

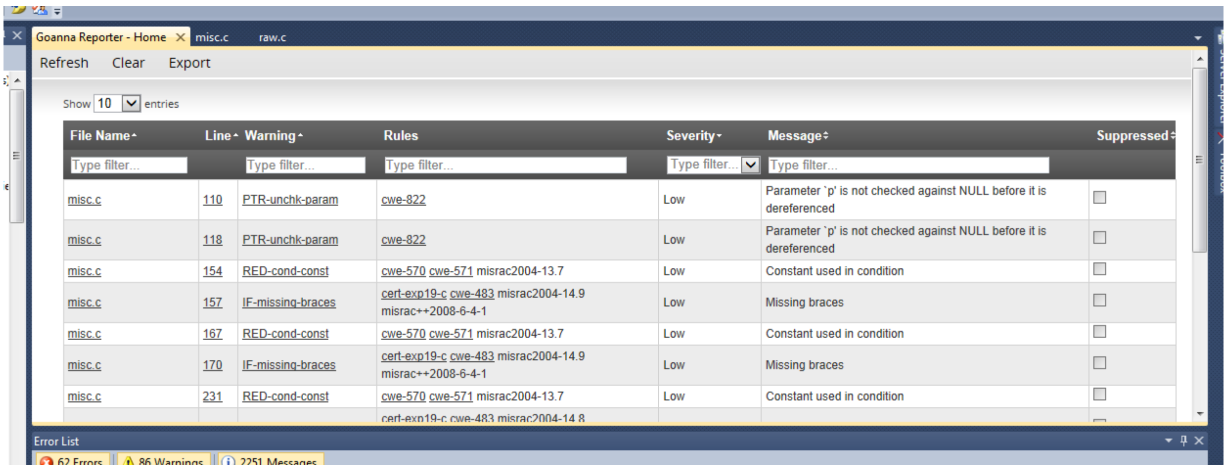
To unsuppress a warning, ensure you are showing suppressed warnings in the Warnings tab. Select the warning you wish to unsuppress, right-click on it, and select Unsuppress Warning from the context menu.

You can unsuppress multiple warnings at once. To do this, select the warnings in the same way as you would to suppress them, right-click on one of the selected warnings, and select Unsuppress Warnings from the context menu.

5.4 Goanna Suppression Manager

The Goanna Suppression Manager enables you to see the status of warnings and select which warnings to suppress or unsuppress using a simple web-based interface.

It allows you to search and sort the list of warnings for a selected project, as well as suppress and unsuppress warnings using the checkboxes on the right hand side of the screen.



File Name	Line	Warning	Rules	Severity	Message	Suppressed?
misc.c	110	PTR-unchk-param	cwe-822	Low	Parameter `p' is not checked against NULL before it is dereferenced	☐
misc.c	118	PTR-unchk-param	cwe-822	Low	Parameter `p' is not checked against NULL before it is dereferenced	☐
misc.c	154	RED-cond-const	cwe-570 cwe-571 misrac2004-13.7	Low	Constant used in condition	☐
misc.c	157	IF-missing-braces	cert-exp19-c cwe-483 misrac2004-14.9 misrac++2008-6-4-1	Low	Missing braces	☐
misc.c	167	RED-cond-const	cwe-570 cwe-571 misrac2004-13.7	Low	Constant used in condition	☐
misc.c	170	IF-missing-braces	cert-exp19-c cwe-483 misrac2004-14.9 misrac++2008-6-4-1	Low	Missing braces	☐
misc.c	231	RED-cond-const	cwe-570 cwe-571 misrac2004-13.7	Low	Constant used in condition	☐
			cert-exp19-c cwe-483 misrac2004-14.8			

5.4.1 Toggling Suppression Status

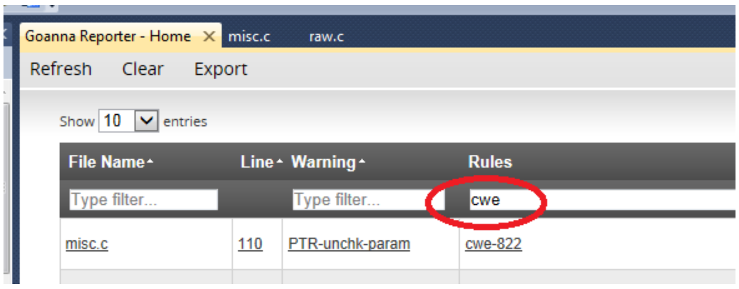
Under the Suppressed? column, you can check/uncheck the display status of a warning. Suppressed warnings will not be displayed in the future Goanna analysis (see 5.1 for details).

5.4.2 Searching and Sorting Warnings

The Goanna Suppression Manager provides powerful searching and sorting capabilities for warnings.

Clicking the column headers sorts the results alternately in ascending or descending order.

Each column also has a search box that is specific to the column. Search boxes accept any plain text keyword; for example, to search for all warnings from the check `SPC-return`, type `SPC-return` in the search box on the Warning column.



File Name	Line	Warning	Rules
Type filter...		Type filter...	cwe
misc.c	110	PTR-unchk-param	cwe-822

Clicking a check name in the Warning column will display the detailed description of that check. Additionally, some rules in the Rules column have links to the corresponding rule pages.

If you click file name or a line number, the corresponding file opens in the Visual Studio at that line.

5.4.3 Other Features

Warnings can be exported in CSV format via the Export button.

If you make changes to the source files and then run analysis, new warnings will not automatically appear in the Gonanna Suppresion Manager. If this happens, click Refresh button to reload the warnings.

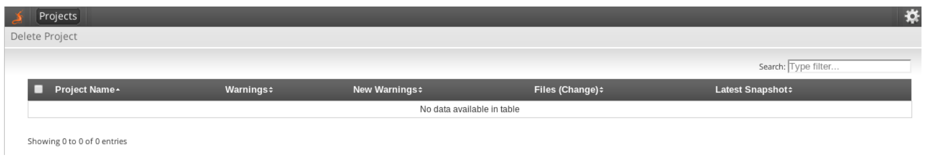
Clear button will delete all suppression, warnings and analysis information for that project. Solution and project options will remain.

6 Goanna Dashboard

Goanna Dashboard allows you to store and visualise the history of your Goanna results. It includes a web server, `goreporter`, used to display these results in a web browser. Each of your analysis runs is captured in a snapshot, which comprises of the warnings for that run as well as the source code analysed in the run. These snapshots are used to track the history of your project in the Goanna Dashboard.

6.1 Getting to the Goanna Dashboard

Click `Goanna Dashboard` in the Goanna menu to open the Goanna Dashboard. This will load the Project Page showing all projects that you have analysed with Goanna from Visual Studio.



Project Name	Warnings	New Warnings	Files (Change)	Latest Snapshot
No data available in table				

Showing 0 to 0 of 0 entries

If you load the Goanna Dashboard without first taking a snapshot you will see an empty table with the text "No data available in table". To use the Goanna Dashboard you must first take a snapshot. (by running analysis, or using the "Take Snapshot" menu item)

6.2 Bug Statuses

The Goanna Dashboard allows you to classify bugs into one of five statuses:

Unclassified This is the default status for when a new warning is added to the Goanna Dashboard.

Ignore Ignore this warning; useful when the warning is valid, but does not require immediate attention.

Analyse This warning need to be investigated further before it can be classified properly.

Fix This warning is a problem that needs to be fixed.

Not a Problem This is warning is not a real bug (false positive).

If you select a warning to be either **Ignore** or **Not a Problem**, then these warnings are automatically *suppressed*; Goanna will then ignore these warnings in future analysis runs.

6.3 Severity

Every check in Goanna has been assigned a static severity of either; High, Medium or Low. These severities are represented throughout the Goanna Dashboard in three colours (or a blend of these three colours):

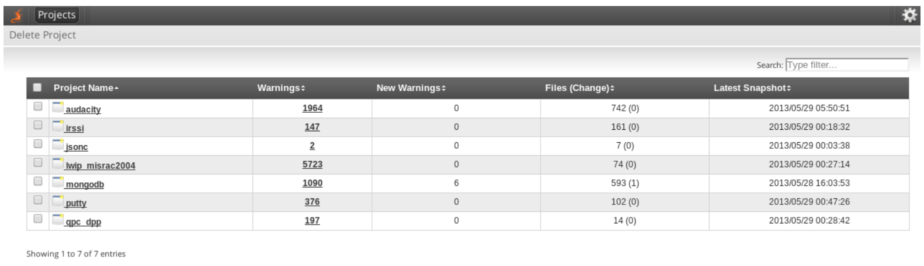
- **High:** Red
- **Medium:** Yellow
- **Low:** Green

6.4 Dashboard Views

6.4.1 Project Page

The project page is usually where you begin when using the Goanna Dashboard. It gives an overview of all projects Goanna has analysed and taken a snapshot of. The table provides a high level overview of the current state of all projects, including the number of warnings, number of new warnings, number of files analysed and the last time a snapshot was taken.

If you do not want to have a project in your dashboard anymore, you can use the checkboxes on the left hand side of the table to select the project(s) you wish to delete, then use the Delete Project button in the Dashboard's toolbar.



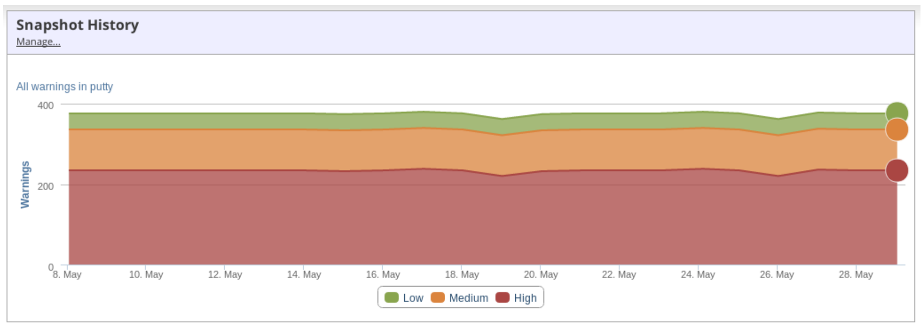
Project Name	Warnings	New Warnings	Files (Change)	Latest Snapshot
audacity	1964	0	742 (0)	2013/05/29 05:50:51
irssi	147	0	161 (0)	2013/05/29 00:18:32
jsonc	2	0	7 (0)	2013/05/29 00:03:38
lwip_misrac2004	5723	0	74 (0)	2013/05/29 00:27:14
mongodb	1090	6	593 (1)	2013/05/28 16:03:53
putty	376	0	102 (0)	2013/05/29 00:47:26
qpc_dpp	197	0	14 (0)	2013/05/29 00:28:42

Showing 1 to 7 of 7 entries

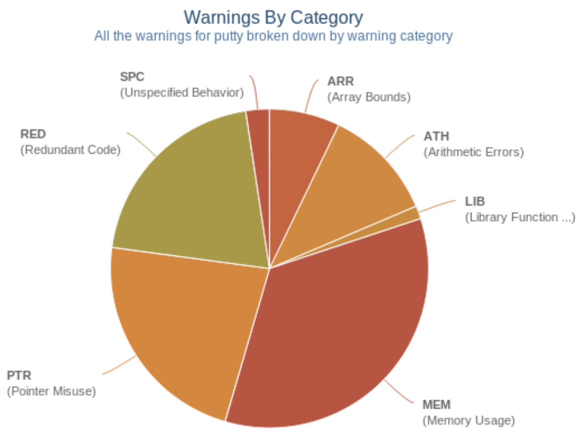
6.4.2 Report Page

Once you select a project you are taken to the project's report page. This page shows four graphs:

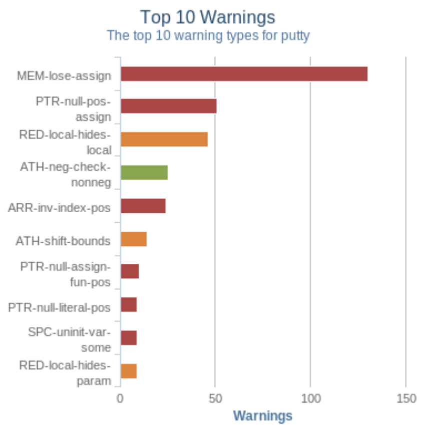
Snapshot History shows the overall progress of your project over time. Each point on the graph is one of your previous snapshots, showing the total number of warnings by severity. Clicking on any point in the graph will change the report page to show details about warnings in that snapshot. By default, the most recent snapshot is selected.



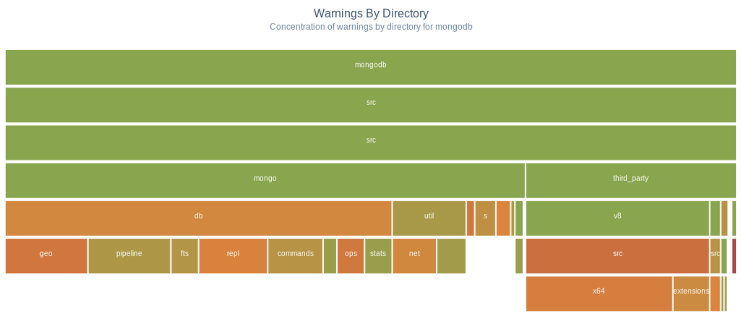
Warnings By Category shows all warnings for your snapshot broken down into each warning type. Clicking on a wedge will show the break down of warnings of that category. Clicking on a subsequent wedge will take you to the warnings browser filtered for that particular warning type.



Top 10 Warnings shows the top ten warning types in this snapshot. Clicking on one of the bars in the chart will take you to the warnings browser filtered by that warning type.

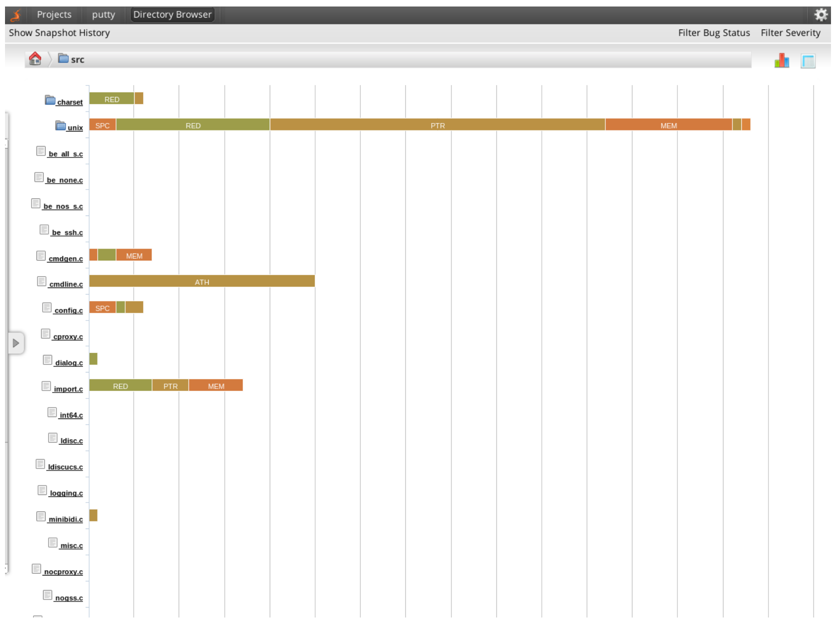


Warnings By Directory shows the concentration of warnings in your directory structure. A red node means that there are more than 10 warnings per file (average) in the directory, a green one means there are zero warnings per file (average). Clicking on a node will load the directory browser in that folder.



Global filters can be applied from the toolbar to filter by warning severity and/or bug status. These filters apply to all charts.

6.4.3 Directory Browser

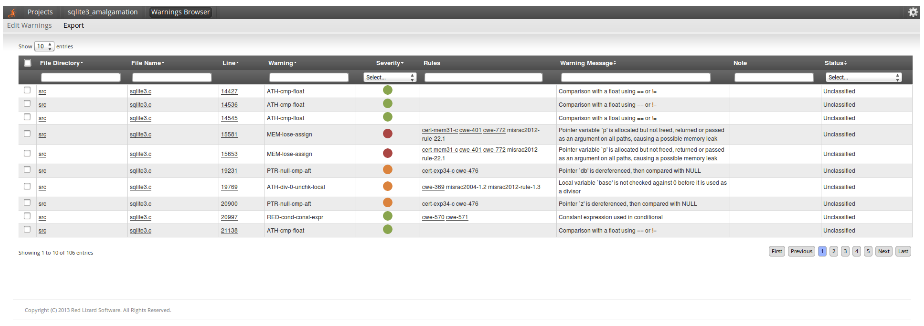


The directory browser is a way to browse through your project's directory structure to see what files and folders have what warnings. Total number of warnings per file or folder are broken down into warning categories and displayed in a bar chart.

The directory browser allows you to browse through your source tree in a few ways. The location bar above the chart allows you to see the path to your current location. Clicking on an item in the location bar will take you there in the directory browser. Similarly, clicking on a directory name in the chart will reload the directory browser with the contents of that folder. To view the contents of a source file, click on its name to load the code browser. To see the details about a particular warning category for a file, click on the segment for that category in the bar of the file or folder to load the warnings browser filtered for your selection.

The chart can be filtered in two ways. The sidebar allows for particular warning types, or warning categories to be turned on and off. In addition, the global filters in the toolbar (severity and bug status) also apply to this chart.

6.4.4 Warnings Browser



Show 10 entries

	File Directory	File Name	Line	Warning	Severity	Rules	Warning Message	Note	Status
☐	src	sqlite3.c	14427	ATH-cmp-float	🟢		Comparison with a float using == or !=		Unclassified
☐	src	sqlite3.c	14536	ATH-cmp-float	🟢		Comparison with a float using == or !=		Unclassified
☐	src	sqlite3.c	14545	ATH-cmp-float	🟢		Comparison with a float using == or !=		Unclassified
☐	src	sqlite3.c	15581	MEM-lose-assign	🔴	cert-mem31-c cwe-401 cwe-772 misrac2012-rule-22.1	Pointer variable `p' is allocated but not freed, returned or passed as an argument on all paths, causing a possible memory leak		Unclassified
☐	src	sqlite3.c	15653	MEM-lose-assign	🔴	cert-mem31-c cwe-401 cwe-772 misrac2012-rule-22.1	Pointer variable `p' is allocated but not freed, returned or passed as an argument on all paths, causing a possible memory leak		Unclassified
☐	src	sqlite3.c	19231	PTR-null-cmp-aft	🟠	cert-exp34-c cwe-476	Pointer `db' is dereferenced, then compared with NULL		Unclassified
☐	src	sqlite3.c	19769	ATH-div-0-unchk-local	🟠	cwe-369 misrac2004-1.2 misrac2012-rule-1.3	Local variable `base' is not checked against 0 before it is used as a divisor		Unclassified
☐	src	sqlite3.c	20900	PTR-null-cmp-aft	🟠	cert-exp34-c cwe-476	Pointer `z' is dereferenced, then compared with NULL		Unclassified
☐	src	sqlite3.c	20997	RED-cond-const-expr	🟢	cwe-570 cwe-571	Constant expression used in conditional		Unclassified
☐	src	sqlite3.c	21138	ATH-cmp-float	🟢		Comparison with a float using == or !=		Unclassified

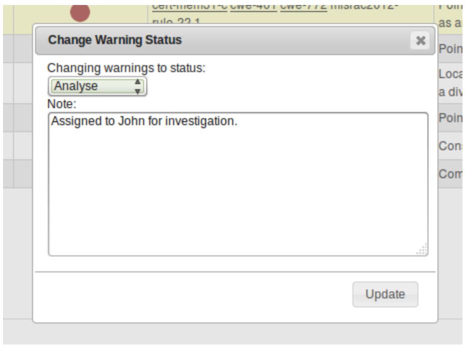
Showing 1 to 10 of 106 entries

Copyright (C) 2013 Red Lizard Software. All Rights Reserved.

The warnings browser shows details of all the warnings in your project. Filtering is possible through the filter boxes in the header of the warnings table. The arrows in the table header allow for sorting.

Clicking on a directory name will take you to the directory browser for that directory. A file name or line number will take you to the source code browser for that file and warning. Clicking on a Rule or Warning name will give you a description of that rule.

Selecting warnings then clicking "Edit Warnings" button opens a dialog where you can change their status and also add a note to the warnings. Clicking `Update` saves these changes.

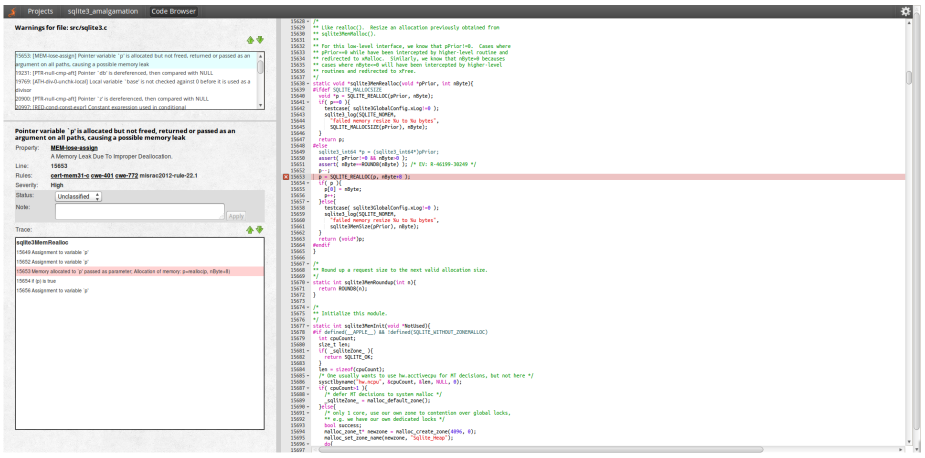


Changing warnings to status: Analyse

Note: Assigned to John for investigation.

The ‘Export’ button allows you to export all warnings or visible warnings to a CSV file.

6.4.5 Code Browser



Warnings for file: src/sqlite3.c

15653: [MEM-lose-assign] Pointer variable `p' is allocated but not freed, returned or passed as an argument on all paths, causing a possible memory leak
19231: [PTR-null-cmp-aft] Pointer `db' is dereferenced, then compared with NULL
19769: [ATH-div-0-unchk-local] Local variable `base' is not checked against 0 before it is used as a divisor
20900: [PTR-null-cmp-aft] Pointer `z' is dereferenced, then compared with NULL
20997: [RED-cond-const-expr] Constant expression used in conditional

Pointer variable `p' is allocated but not freed, returned or passed as an argument on all paths, causing a possible memory leak

Property:	MEM-lose-assign A Memory Leak Due To Improper Deallocation.
Line:	15653
Rules:	cert-mem31-c cwe-401 cwe-772 misrac2012-rule-22.1
Severity:	High
Status:	Unclassified
Note:	

Trace:

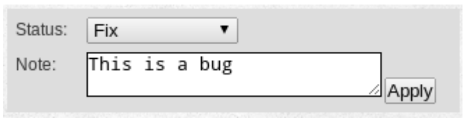
sqlite3MemRealloc
15649 Assignment to variable `p'
15652 Assignment to variable `p'
15653 Memory allocated to `p' passed as parameter; Allocation of memory: p=realloc(p, nByte+8)
15654 if (p) is true
15656 Assignment to variable `p'

```c
/*
** Like realloc().  Resize an allocation previously obtained from
** sqlite3MemMalloc().
**
** For this low-level interface, we know that pPrior!=0.  Cases where
** pPrior==0 while have been intercepted by higher-level routine and
** redirected to xMalloc.  Similarly, we know that nByte>0 becauses
** cases where nByte<=0 will have been intercepted by higher-level
** routines and redirected to xFree.
*/
static void *sqlite3MemRealloc(void *pPrior, int nByte){
#ifdef SQLITE_MALLOCSIZE
  void *p = SQLITE_REALLOC(pPrior, nByte);
  if( p==0 ){
    testcase( sqlite3GlobalConfig.xLog!=0 );
    sqlite3_log(SQLITE_NOMEM,
      "failed memory resize %u to %u bytes",
      SQLITE_MALLOCSIZE(pPrior), nByte);
  }
  return p;
#else
  sqlite3_int64 *p = (sqlite3_int64*)pPrior;
  assert( pPrior!=0 && nByte>0 );
  assert( nByte==ROUND8(nByte) ); /* EV: R-46199-30249 */
  p--;
  p = SQLITE_REALLOC(p, nByte+8 );
  if( p ){
    p[0] = nByte;
    p++;
  }else{
    testcase( sqlite3GlobalConfig.xLog!=0 );
    sqlite3_log(SQLITE_NOMEM,
      "failed memory resize %u to %u bytes",
      sqlite3MemSize(pPrior), nByte);
  }
  return (void*)p;
#endif
}

/*
** Round up a request size to the next valid allocation size.
*/
static int sqlite3MemRoundup(int n){
  return ROUND8(n);
}

/*
** Initialize this module.
*/
static int sqlite3MemInit(void *NotUsed){
#if defined(__APPLE__) && !defined(SQLITE_WITHOUT_ZONEMALLOC)
  int cpuCount;
  size_t len;
  if( _sqliteZone_ ){
    return SQLITE_OK;
  }
  len = sizeof(cpuCount);
  /* One usually wants to use hw.acctivecpu for MT decisions, but not here */
  sysctlbyname("hw.ncpu", &cpuCount, &len, NULL, 0);
  if( cpuCount>1 ){
    /* defer MT decisions to system malloc */
    _sqliteZone_ = malloc_default_zone();
  }else{
    /* only 1 core, use our own zone to contention over global locks,
    ** e.g. we have our own dedicated locks */
    bool success;
    malloc_zone_t* newzone = malloc_create_zone(4096, 0);
    malloc_set_zone_name(newzone, "Sqlite_Heap");
    do{
```

The code browser displays a file and all its Goanna warnings. The right hand side shows the source code of the file currently opened with warnings highlighted in three colours based on severity. The left hand side has the details of warnings for this file. The box at the top of the left hand pane allows you to select a warning to see in more detail.

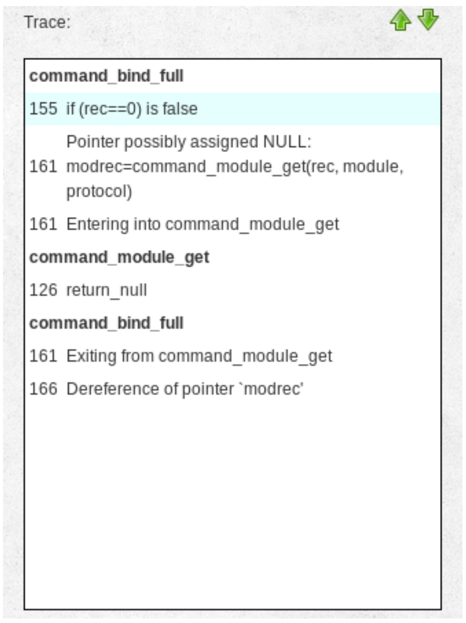
Like in the warnings browser, you are able to change the status and add a note to each warning in the code browser. This can be done by changing the status and/or adding a note and pressing Apply.

You can also click the warning header at the top of the left hand pane to jump to the corresponding line in Visual Studio.



Status:	Fix
Note:	This is a bug

If there is trace information for a warning it will also appear in this pane. You can step through the trace just like a debugger. To go to a step in the trace, click on it and the source code browser will jump to the corresponding line. You can then navigate through the trace using either the up/down arrows on top of the trace dialog, or by using the up/down arrow keys on your keyboard.



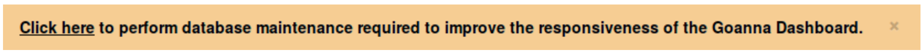
Trace:

command_bind_full
155 if (rec==0) is false
Pointer possibly assigned NULL:
161 modrec=command_module_get(rec, module, protocol)
161 Entering into command_module_get
command_module_get
126 return_null
command_bind_full
161 Exiting from command_module_get
166 Dereference of pointer `modrec'

6.5 Database Upgrades

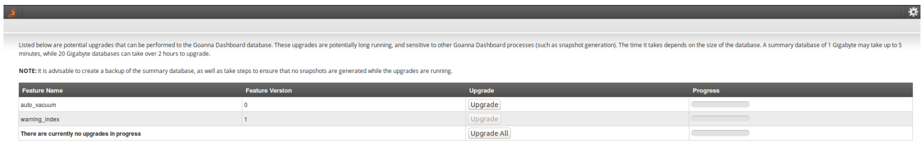
If you have upgraded from an older version of Goanna, Goanna may offer you to perform optional “database upgrades“ to improve the performance of the Goanna Dashboard.

When you open Goanna Dashboard with the existing database, you may see the following notification:



Click here to perform database maintenance required to improve the responsiveness of the Goanna Dashboard. ×

Clicking the link will show a list of available optional upgrades:



Listed below are potential upgrades that can be performed to the Goanna Dashboard database. These upgrades are potentially long running, and sensitive to other Goanna Dashboard processes (such as snapshot generation). The time it takes depends on the size of the database. A summary database of 1 Gigabyte may take up to 5 minutes, while 20 Gigabyte databases can take over 2 hours to upgrade.

NOTE: It is advisable to create a backup of the summary database, as well as take steps to ensure that no snapshots are generated while the upgrades are running.

Feature Name	Feature Version	Upgrade	Progress
auto_vacuum	0	Upgrade	
warning_index	1	Upgrade	
There are currently no upgrades in progress		Upgrade All	

In this version of Goanna, the following optional upgrades (called *features*) may be available:

- **`auto_vacuum`**: In Goanna 3.2.4 and before, deleting a snapshot or a project did not immediately delete corresponding data from the database. This may result in the database size to never shrink.
  Performing this upgrade causes Goanna to delete all residual data from removed snapshots and projects, and set up the database so that when a snapshot or a project is removed, Goanna removes corresponding data immediately. This ensures that the database size is always minimal.
  This upgrade was introduced in Goanna 3.3.0.
- **`warning_index`**: Performing this upgrade causes Goanna to apply optimizations to database indexes, resulting in improved performance, especially when loading Warnings Browser view.
  This upgrade was introduced in Goanna 3.3.0.

To apply an upgrade, click “Upgrade” button next to a desired feature. Alternatively, clicking “Upgrade All” button will apply all available upgrades.

Important Notes

We recommend that you take a backup of the database before applying any of the upgrade. The database is located in the following location:

`<default-project-location>\summary.goannadb`

For Visual Studio, the default location is:

- On Windows XP and Windows Server 2003:
  `C:\Documents and Settings\<name-of-user-who-installed-goanna>\My Documents\Visual Studio <version-of-visual-studio>\Projects\summary.goannadb`
- On Windows Vista, Windows Server 2008, and all later versions of Windows:
  `C:\Users\<name-of-user-who-installed-goanna>\Documents\Visual Studio <version-of-visual-studio>\Projects\summary.goannadb`

For Atmel Studio, the default location is:

- On Windows XP and Windows Server 2003:
  `C:\Documents and Settings\<name-of-user-who-installed-goanna>\My Documents\Atmel Studio\<version-of-atmel-studio>\summary.goannadb`
- On Windows Vista, Windows Server 2008, and all later versions of Windows:
  `C:\Users\<name-of-user-who-installed-goanna>\Documents\Atmel Studio\<version-of-atmel-studio>\summary.goannadb`

You should not run Goanna analysis, interact with the Goanna Dashboard, or otherwise run any Goanna commands, until the upgrade is complete.

Performing upgrade will take a long time depending on the size of the database. If your database is significantly large, this may take hours to complete.

6.6 Project Settings (Advanced)

In general, the Goanna Dashboard should work for your project out of the box. However, there are a few advanced settings available if you want to customise the behaviour.

Settings are applied on a per-project basis. To access your project's settings click the gear menu in the top right hand corner after loading your project (through the project page), and select Project Settings.

6.6.1 Code Browser Character Encodings

By default the Dashboard will recognize source files in ASCII and UTF-8 (and additionally on Windows, the default character encoding used by your system). If your source files are not in any of these encodings, you will need to specify the encoding here in order for it to display correctly in the source code browser. A link is available in the Project Settings window listing all the encodings supported.

7 Command Line Options

This section is a detailed reference of the command line options that can be added to the *Extra arguments* field in your Project or Solution Settings.

7.1 Options

`--absolute-path` Print absolute paths in warnings.

`--all-checks` Run all available checks (overrides all other check related options).

`--brief-trace` Show immediately relevant decisions in trace output, not the majority of decisions.

`--c++` Indicate that file(s) contain C++ code.

`--check=`*<name>* Run a specific check (overrides any checks file).

`--checks-file=`*<file>* Use the checks listed in *<file>* instead of the default checks in `properties.init` file.

`--checks=`*<standard>* Run all checks in the specified coding standard. For example, `--checks=misrac2004` runs all available checks in the MISRA C:2004 standard.

`--checks=`*<standard>-<rule>* Run the check(s) corresponding to one rule in the specified coding standard. For example, `--checks=misrac2004-12.8` runs the check(s) that implement MISRA C:2004 rule 12.8.

`--color, --colour` Only available on Linux. Output in color.

`--columns` Print column positions in warnings.

`--db=`*<file>* Specify the database file to use for persistent information.

`--directory=`*<dir>* Before doing anything, change to *<dir>*.

`--force-analysis` Re-analyze files that have not changed since last run.

`--help` Print help message for common options.

`--ignore-errors` Ignore errors from the compiler.

`--license-server=`*<server[:port]>* Attempt to contact a license server at address *<server>*. *<Port>* is optional (defaults to 6200).

`--output-checks` Output the checks that are currently loaded.

`--output-format=`*<format>* Specify a warning format used by Goanna to output warnings. The following special strings in *<format>* are expanded:

- **`%FILENAME%`** the filename
- **`%RELFILE%`** the filepath and filename
- **`%RELPATH%`** the filepath
- **`%ABSFILE%`** the absolute filepath and filename
- **`%ABSPATH%`** the absolute filepath
- **`%DBRELFILE%`** the filepath relative to the database file and filename
- **`%DBRELPATH%`** the filepath relative to the database file
- **`%LINENO%`** the line number
- **`%COLUMN%`** the column number
- **`%CHECKNAME%`** the check identifier
- **`%SEVERITY%`** the checks severity rating
- **`%MESSAGE%`** the warning message
- **`%RULES%`** corresponding rule(s) from coding standards, if any
- **`%TRACE%`** counter example if any
- **`%FUNCTION%`** the function name
- **`%SUPPRESSED%`** a `*` if the warning is suppressed
- **`%WARNINGID%`** the hash of this warning
- **`%EOL%`** a line break
- **`%%`** a literal %.

The default warning format is: "`%RELFILE%:%LINENO%:  warning:  Goanna[%CHECKNAME%] Severity-%SEVERITY%, %MESSAGE%.  %RULES%%EOL%`".

`--project-dir=`*<path>* Specify the location of the project root; default is `..`

If the `--project-dir` option is given, then `--goanna-dir` is also required.

`--quiet` Only display warnings and no other output.

`--suppress=`*<warning id>* Suppress warning *<warning id>*.

`--suppression-status` Output suppression status markers, without suppressing warnings.

`--system-headers` Process system header files. Default: no.

`--timeout=`*<n>* Set a timeout (in seconds) for analysis of each source file. Default: 240.

Important: Setting this value to 0 (meaning infinite) is discouraged; this may cause Goanna to not terminate!

`--trace` Prints out a trace through the function that leads to the warning. This is helpful for understanding why the warning occurs.

`--trace-format=`*<format>* Specify the format to output traces. The following special strings are used in the trace format:

`%FILENAME%` the filename

`%RELFILE%` the filepath and filename

`%RELFILEX%` the filepath and filename followed by "`:`", or blank if in the current source file

`%RELPATH%` the filepath

`%ABSFILE%` the absolute filepath and filename

`%ABSFILEX%` the absolute filepath and filename followed by "`:`", or blank if in the current source file

`%ABSPATH%` the absolute filepath

`%DBRELFILE%` the filepath relative to the database file and filename

`%DBRELPATH%` the filepath relative to the database file

`%FUNCTION%` the function name

`%LINE%` the line number

`%TEXT%` text describing the event on the trace

`%TYPE%` the type of the trace line

`%EOL%` a line break

`%%` a literal %.

The default trace format is: "`%LINE%:  %TYPE% - %TEXT%%EOL%`".

`--unsuppress=`*<warning id>* Unsuppress warning *<warning id>*.

`--user-headers` Process user header files. Default: no.

`--verbose` Display additional output information.

`--version` Print version information.

`--warning-ids` Output warning hashes.

`--with-cc=`*<compiler>* Specify the C compiler executable to run (if `--nc` is not specified). Also affects the default dialect when no `--dialect` is specified.

`--with-cxx=`*<compiler>* Specify the C++ compiler executable to run (if `--nc` is not specified). Also affects the default dialect when no `--dialect` is specified.

Advanced Command Line Options

The following options are intended to be used only in cases where your environment requires them. In general, you do not need to use these options.

`--32` Analyze code for 32-bit targets (longs and pointers are 32 bits wide).

`--64` Analyze code for 64-bit targets (longs and pointers are 64 bits wide).

`--advanced-help` Print help message for advanced options.

`--dialect=`*<file>* Specify the dialect of C/C++ compilers. Available dialects are:

- `gnu` (available on Atmel Studio only)
- `microsoft`

If you use this option, you should also specify `--with-cc`, `--with-cxx` and/or `--with-ld` to specify the paths to the compiler(s) and linker. If these are not specified, then Goanna will assume the default name for the specified dialect, which may not be what is available on your system.

If none of `--dialect`, `--with-cc`, `--with-cxx` or `--with-ld` are specified, then Goanna will assume the default of `gnu` dialect with `gcc` C compiler, `g++` C++ compiler and `ld` linker.

`--error` Exit with error status code when warnings emitted.

`--exclude=`*<file>* Exclude the specified *<file>* from analysis.

`--html-report=`*<output type>* After analysis, also generate analysis report files in HTML format. You can optionally specify type of HTML reports to be generated ('summary', 'warnings' or 'all'). `--html-report` with no type will generate all available reports.

`--input-encoding=`*<type>* Specify the character encoding of the source file:

us-ascii ASCII (default)

utf-8 UTF-8

ansi (Available on Windows only) default character encoding of the system

`--internal-error=`*<value>* Exit with *<value>* on internal error.

`--ipa-iterations=`*<value>* Specify the number of times interprocedural analysis iterates towards a fixed point. The default is 2.

Important: Setting this value to 0 (meaning keep iterating until a fixed point is reached) is discouraged; this may cause Goanna to not terminate!

`--ipa-trace-depth=`*<value>* How many levels of inlining are performed for interprocedural traces. Default: 5.

Important: Setting this value to -1 (meaning infinite) is discouraged; this may cause Goanna to not terminate!

`--issue-report=`*<type>* Control generation of issue report files:

never Never

on-failure On failures only

on-error On failures and analysis errors

timeout On failures, errors, and timeouts

always Always (even if successful)

`--license-borrow-hours=`*<number>* When contacting license server, borrow license for *<number>* of hours. Defaults to 1, maximum of 24.

`--license-dir=`*<directory>* Set directory in which to look for a license file.

`--no-globals` Do not analyze global integer variables.

`--no-ipa` Disable interprocedural analysis.

`--output-file=`*<file>* Append warning messages to a specified file.

`--output-spec=`*<file>* Use the contents of *<file>* as the output-format.

`--output-xml=`*<file>* After analysis, also output warnings in XML format to *<file>*.

`--parse-error-log=`*<file>* Log parse errors to the specified file instead of stderr.

`--timeout-error=`*<value>* Exit with status code *<value>* when too many timeouts occur.

`--timeout-limit=`*<value>* Maximum number of per-phase timeouts. Default: 3.

Important: Setting this value to 0 (meaning infinite) is discouraged; this may cause Goanna to not terminate!

`--timeout-per-phase=`*<n>* Set a timeout (in seconds) for each phase of analysis. This is useful if you have a few functions that take very long to analyze and you would like to limit the time spent on these, while still getting as many results as possible on everything else. Default: 60.

Important: Setting this value to 0 (meaning infinite) is discouraged; this may cause Goanna to not terminate!

Diagnostics Command Line Options

The following options are provided for diagnostics purposes only. **Do not use these options unless directed by Red Lizard Software support team.**

`--alias`

`--configure=`*<dialect>*

`--dialect-mod=`*<dialect-mod>*

`--dataflow`

`--diagnostics-mode`

`--no-alias`

`--no-dataflow`

`--no-default-packages`

`--package=`*<package>*

`--package-dir=`*<directory>*

`--record`

Any unrecognized options are treated as compiler arguments.

8 Troubleshooting

This section describes a number of known issues and solutions to them. Should you find more issues you think we should know about, please contact us at mailto:support@redlizards.com.

8.1 Known Issues For Visual Studio Users

Note: The following issues do not apply for Atmel Studio users.

Managed Code

Managed C++, C++/CLI syntax extensions, and CLR (Common Language Runtime) related compiler options are not supported.

Generated Files

Goanna Studio cannot distinguish between generated files and the hand-coded files that generate them, and will issue warnings in the files in which they are found, in every instance they are found.

This means that large generated files may yield a large number of warnings stemming from a single bug in the hand-coded file which generated it. This will cause a warning for every manifestation of this bug in the generated code.

The `#import` Directive

When using the `#import` directive to import a dynamic library (`.dll`) or type library (`.tlb`), it is necessary to compile the file containing the directive before it is analysed by Goanna Studio. Compilation generates a header file with the extension `.tlh`.

Then in the source code where you perform your import, you can use the `_GOANNA` macro to conditionally include the `.tlh` file. For example:

```
#ifdef _GOANNA
#include ".\Debug\msxml3.tlh"
#else
#import "msxml3.dll"
#endif
```

Precompiled Header Through Files

Goanna Studio does not currently handle 'precompiled header through' files, used with the MSVC option `/Yu`, if the specified file does not exist. The third-party C/C++ parser we use does not handle precompiled header files in the way given by the `/Yu` option.

Target Platforms (Visual Studio 2005 & 2008 only)

Only the following target platforms are supported: `Win32`, `x86`, `x64`, `ia64`, `ARM`, `Pocket PC 2003 (ARMV4)`, `Smartphone 2003 (ARMV4)`, `STANDARDSDK_500 (MIPSIV_FP)`, `Windows Mobile 5.0 Pocket PC SDK (ARMV4I)` and `Windows Mobile 5.0 Smartphone SDK (ARMV4I)`.

Important: The name of target platforms must match exactly as in the above list.

You may get a parse error if your project targets a platform not in this list.

Performance With Many Warnings

If you enable many checks, or use checks in `misrac2004`, `misrac++2008` or `misrac2012` checks packages, then Goanna may display many warnings (1,000 or over) into Visual Studio or Atmel Studio's "Error List" window. If you select all types of errors (Errors, Warnings and Messages) to be displayed, this may slow down Goanna analysis and degrade performance of the IDE substantially.

8.2 Support

Some unforeseen issues might occur while using Goanna Studio, depending on your particular compiler version, include files, hardware, undocumented features in C/C++ or some combination of these. If you come across any problems, please contact mailto:support@redlizards.com or visit out support page at http://redlizards.com/resources/.

Index