ECLAIR Frequently Asked Questions

Contents

1 Introduction **2**

2 ECLAIR **3**
- 2.1 What is ECLAIR? . . . 3
- 2.2 What is an ECLAIR package? . . . 3
- 2.3 So I can buy licenses for the packages, not for the entire ECLAIR platform? . . . 3
- 2.4 Do you offer trial versions? . . . 4
- 2.5 What is the meaning of ECLAIR version numbers? . . . 4

3 Toolchains **4**
- 3.1 Why do you want to know which toolchain(s) am I using? . . . 5
- 3.2 What are the toolchains supported right now? . . . 5
- 3.3 If I tell you that I use toolchain X, will I only obtain ECLAIR for toolchain X? . . . 5
- 3.4 What happens if the toolchain I use is not supported? . . . 5
- 3.5 Why only few toolchains are listed in my contract/license? . . . 5
- 3.6 How can I make sure my toolchain is properly intercepted by ECLAIR? . . . 6

4 Licensing **6**
- 4.1 What are the parameters defining ECLAIR license models? . . . 6
- 4.2 Upon which conditions can a per-project license be granted? . . . 7
- 4.3 What are the factors influencing the cost of per-seat licenses? . . . 7
- 4.4 So, coverage and license-enforcing mechanisms do not influence the price? . . . 7
- 4.5 What is the advantage of site/multisite/enterprise licenses then? . . . 8
- 4.6 What are the available license-enforcing mechanisms? . . . 8
- 4.7 What are the pros and cons of node-locked keys? . . . 8
- 4.8 What are the pros and cons of dongle-locked keys? . . . 8
- 4.9 What are detachable keys? . . . 9
- 4.10 What are the pros and cons of detachable keys? . . . 9
- 4.11 Who chooses the license enforcing mechanism(s)? . . . 9

4.12 How many pools of detachable licenses should I have, and how large? . . . 10
4.13 What are the ECLAIR licensing requirements for Jenkins? . . . 10
4.14 How do I get rid of expired trial licenses? . . . 11

5 Maintenance **11**
5.1 Is maintenance included in the ECLAIR license? . . . 12
5.2 Am I compelled to renew maintenance? . . . 12
5.3 Can I keep using the issue-tracking system after maintenance expiration? . . . 12
5.4 Can I reinstate maintenance after it has expired? . . . 12
5.5 Why have I to pay, for reinstatement, the maintenance fees of past years? . . . 12

6 ECLAIR Reports **12**
6.1 What are the main categories for ECLAIR outputs? . . . 12
6.2 What are the main output formats? . . . 13
6.3 Which kinds and format of reports are available depending on the license? . . . 13
6.4 Why clicking on a ☞ symbol in the manual results in an error? . . . 13
6.5 Is it possible to generate summaries in .docx format instead of .odt? . . . 13
6.6 Is it possible to generate summaries in PDF? . . . 14
6.7 Should I pay attention to reports of the *error* kind? . . . 14

7 Problems **15**
7.1 Need I Use the Latest Version of Sentinel LDK Run-Time Environment? . . . 15
7.2 I get a Sentinel LDK Protection System error code of the form Hnnnn: why? . . . 15
7.3 Running the analysis in the GUI fails, but it succeeds outside the GUI: why? . . . 17
7.4 A project (possibly an ECLAIR demo project) does not work: why? . . . 17
7.5 ECLAIR seems to be ignoring the code in my project: why? . . . 18
7.6 The ECLAIR Bug Finder gives a false positive: will you fix it? . . . 18

References **19**

1 Introduction

This document answers some of the frequent questions that prospects and customers ask about ECLAIR.

If you have a question that you believe ought to be in this document, please use the contact information provided below.

Note that the PDF version of this document has clickable links.

2 ECLAIR

In this section, we collect frequently asked questions about the ECLAIR system itself.

2.1 What is ECLAIR?

ECLAIR is a general platform for the verification of C and C++ source code based on static analysis. The range of its applications include: automatic check of compliance with respect to coding standards, bug finding, computation of software metrics, automatic test input generations, deep semantic analysis, semantic matching and patching.

2.2 What is an ECLAIR package?

An ECLAIR package is an application running under the ECLAIR platform. A package implements a set of features, and different packages can be combined together. Examples of commercialized ECLAIR packages are:

- MC2: MISRA C:2004 + Essentials[1]
- MC3: MISRA C:2012 + Essentials[2]
- MP1: MISRA C++:2008 + Essentials[3]
- B: BARR-C:2018 + Essentials[4]

All MISRA packages include the B package. The packages can be combined in all possible ways. The ECLAIR combined packages are denoted by the following identifiers:

- MC23: (MC2 + MC3)
- MC2P1: (MC2 + MP1)
- MC3P1: (MC3 + MP1)
- MC23P1: (MC2 + MC3 + MP1)

2.3 So I can buy licenses for the packages, not for the entire ECLAIR platform?

Exactly.

[1] https://www.bugseng.com/sites/default/files/resources/ECLAIR_MC2.pdf
[2] https://www.bugseng.com/sites/default/files/resources/ECLAIR_MC3.pdf
[3] https://www.bugseng.com/sites/default/files/resources/ECLAIR_MP1.pdf
[4] https://www.bugseng.com/sites/default/files/resources/ECLAIR_B.pdf

2.4 Do you offer trial versions?

Yes: visit https://bugseng.com/eclair/free-trial to obtain one.

We also recommend you see the *ECLAIR Evaluation Guide* for an effective evaluation of ECLAIR: you will receive a PDF copy of this document as soon as you request a trial version.

2.5 What is the meaning of ECLAIR version numbers?

Each ECLAIR version is identified by three decimal natural numerals separated by two dots:

major . **minor** . **patch**

where

major is the *major version number*;

minor is the *minor version number*;

patch is the *patch version number*.

An example of complete version number is "3.6.0" (where the double quotes are not part of the version number).

When backwards compatible bug fixes or when backward compatible new functionality is added, the patch version number is incremented.

When changes are made that are not backwards compatible, the minor version number is incremented and the patch version number is set to 0.

The major version number is special: when it changes it denotes a completely different product. If you read your license, you will see that it covers ECLAIR 3.*: this means that, with a valid maintenance contract in place (see *Maintenance* below), you are entitled to receive all minor and patch versions of ECLAIR (major) version 3 subsequent to the one you originally received.

However ECLAIR version 4 will be a completely different product: there will certainly be a migration path for users of ECLAIR version 3 and support for ECLAIR version 3 will continue for some time. All this will be officialized upon release of ECLAIR version 4 and cannot be anticipated now.

3 Toolchains

One of the key features of ECLAIR is that it spares the customers from any configuration concerning the precise C/C++ dialects they are using. A C/C++ dialect is given by the combination of the toolchain (compiler, assembler, librarian, linker: we call these *toolchain components*) being used, along with the options that are used on each toolchain component invocation.

However, the fact that the tool takes into account the precise details of the language dialect(s) in use [1] has an obvious implication: given that the user needs not worry about that, BUGSENG has to do it, and a precise model of each supported toolchain and its dependence on compile-time options must be part of ECLAIR.

3.1 Why do you want to know which toolchain(s) am I using?

Because there are many toolchains out there, a good portion of which is still maintained, so new releases are published on a regular basis. Let us consider GCC, the GNU C/C++ compiler: this is a family of toolchains, each characterized by the precise version, the target architecture and (in the case of open-source toolchains) the modifications that have been performed. So, when we say "GCC" we are referring to hundreds of different toolchains. Summarizing, the number of C/C++ toolchains currently is probably on the order of tens of thousands.

At any given point in time ECLAIR supports many of these toolchains, but not all. We add support for new toolchain releases on a regular basis, giving priority to the most used ones in the embedded system industry. So, to come to the point, we would like to know which toolchain(s) you are using in order to tell you:

- whether it is currently supported or not;
- if it is not, whether we plan to support it and when;
- in any case, we use your input to make sure our priorities are up-to-date with the needs of our customers and prospects.

3.2 What are the toolchains supported right now?

There are hundreds of them, too many to be listed here with full details, from all the major suppliers, including ARM®, CodeWarrior™, Cosmic Software, CrossWorks™, GCC, Green Hills®, HighTec, IAR™, Intel®, Keil Software®, MPLAB®, Microsoft®, QNX™, Renesas Electronics, SOFTUNE™, TASKING®, Texas Instruments™, Wind River®, and clang/LLVM.

3.3 If I tell you that I use toolchain X, will I only obtain ECLAIR for toolchain X?

No: all shipped versions of ECLAIR contain support for all the supported toolchains.

3.4 What happens if the toolchain I use is not supported?

Under the assumption that you inform us, there are two possibilities:

1. Your toolchain is widely used (or is likely to be in the near future) and you are not in a hurry: it will probably be supported in the next ECLAIR release.
2. All the other cases: we will ask you to contribute to the development costs of its model (we are talking about a sum in the range 1-3 kEUR) and to give us a couple of weeks to implement it.

3.5 Why only few toolchains are listed in my contract/license?

Because for those listed in the contract, the *contractually supported toolchains*, you are entitled to premium support, which includes we will have to deal with possible defects concerning such toolchains within a maximum time specified in the contract itself. For all the other toolchains, we will still provide support, but you may have to wait for the following ECLAIR release.

The number of contractually supported toolchains in our offers is a function of the userbase, and it is a matter of pre-contractual negotiation: if you want more toolchains to be contractually supported we

will send you an offer to that effect. Note that toolchains for which support you contributed to the development costs will be automatically added to your list of contractually supported toolchains.

3.6 How can I make sure my toolchain is properly intercepted by ECLAIR?

Look at the *Number of analysis frames* table in the ECLAIR report. Does the number of *UNIT* frames correspond to the number of translation units in your project? Does the number of *PROGRAM* frames correspond to the number of executables your linker has produced? In case of a negative answer, then no, your toolchain has not been properly intercepted by ECLAIR. See Chapter *Intercepting the Toolchain* of the *ECLAIR User's Manual*.

In case the situation is not clear-cut, consult the *Analysis Frames* section in the ECLAIR report: there you will find a complete list of the toolchain components' invocations intercepted by ECLAIR: if you find something missing, check that you have followed the instructions given in Chapter *Intercepting the Toolchain* of the *ECLAIR User's Manual*.

4 Licensing

ECLAIR is proprietary, protected commercial software owned by BUGSENG. In order to use ECLAIR you need a license from BUGSENG. However, as the word *license* is ambiguous, we will distinguish:

License agreement: The contract allowing a customer to use ECLAIR with certain limitations and upon certain conditions.

License model: A combination of parameters that defines how a customer can use ECLAIR. This combination is reflected into the license agreement.

License-enforcing mechanism: A combination of technical measures that *assist* in checking compliance with the license agreement.

Please make sure you appreciate the distinction between these concepts. For instance, the fact that the license-enforcing mechanism does not block a certain use of ECLAIR does not imply that such use complies to the license agreement.

4.1 What are the parameters defining ECLAIR license models?

They are the following:

Userbase: The criterion defining the group of users that are allowed to use ECLAIR. There are the following possibilities:

- **Per-seat:** The userbase is defined by the number of individually-named users who can use any component of ECLAIR, possibly adjusted to take into account the requirements of Jenkins controllers and agents (see "*What are the ECLAIR licensing requirements for Jenkins?*" for more details). BUGSENG does not require customers to communicate the identities of users, except for the users that use BUGSENG support services (see "*Maintenance*" for more details).
- **Per-project:** The userbase is defined indirectly as the set of all users working on a specific project for a specific purpose.

Term: The criterion defining for how long the userbase is allowed to use ECLAIR. There are the following possibilities:

Indefinite-term: Use is allowed for an indefinite amount of time as long as the customer complies to the license agreement. Indefinite-term licenses include one year of support services including updates; such services can be optionally bought at a predetermined price for subsequent years (see "*Maintenance*" for more details).

Subscription-based: Use is allowed for one-year time periods. Subscription-based licenses are automatically renewed every year unless they are terminated by one of the parties; they always include support services with updates.

Coverage: An attribute describing the relationship between the licensed userbase and the customer potential userbase. There are the following possibilities:

Site: The userbase comprises all the potential ECLAIR users of the customer (e.g., all software developer and all quality-assurance people) at a given site.

Multisite: The userbase comprises all the potential ECLAIR users of the customer at a given set of sites. In addition, there is a single point of contact for all contractual actions, including releases, updates, technical support and payments.

Enterprise: Like site or multisite when the sites covered are the totality of customer sites.

Partial: All other cases.

4.2 Upon which conditions can a per-project license be granted?

This is decided on a case-by-case basis, but typically the project should be not-for-profit (independently from the nature of the organizations involved in the project).

4.3 What are the factors influencing the cost of per-seat licenses?

There are only two factors:

1. The cardinality of the userbase for which the license is bought at once. BUGSENG operates a volume discount policy: the higher is the number of users, the lower is the cost per user.
2. The licensed features: the price increases as the number of licensed features increases. Some features are bundled with others with no price increase: for instance, the *ECLAIR B* package is bundled with all MISRA packages with no price increase, even though the price for *ECLAIR B*, when bought in isolation, is not zero.

4.4 So, coverage and license-enforcing mechanisms do not influence the price?

Exactly: an enterprise license and a partial license will cost the same if the licensed features are the same and the userbases have the same cardinality.

4.5 What is the advantage of site/multisite/enterprise licenses then?

Such licenses allows the generation of detailed reports in textual, HTML and XML format and their distribution within the site/multisite/enterprise. For instance, HTML reports can be published on the internal LAN and consulted by anyone having access to the LAN using any supported web browser.

In contrast, with a partial license detailed reports can only be consulted using the ECLAIR report browser. Summary reports can be generated in any supported format with any license.

4.6 What are the available license-enforcing mechanisms?

There are three license-enforcing mechanisms. What is common to them is that running ECLAIR requires a *key*:

Node-locked: The key is fully implemented in software and is locked to a specific machine. Remote access is not allowed. Transferring a node-locked key, a.k.a. *rehosting*, is possible with BUGSENG intervention. One node-locked key serves one user.

Dongle-locked: The key resides on an USB dongle, which can be moved to different machines. Remote access is not allowed. One dongle-locked key serves two users.

Detachable: The key is implemented in software and can be detached by a (possibly remote) license server. One detachable license serves three users.

4.7 What are the pros and cons of node-locked keys?

Pros:

1. They are fast: all communications occur within the same machine.
2. The user can use ECLAIR on that machine at any time.

Cons:

1. The user can only use ECLAIR on that machine.
2. If the machine crashes badly, the node-locked key can be lost. It is thus recommended to use hard disk drives supporting SMART (Self-Monitoring, Analysis and Reporting Technology) and to rehost the key at the first sign of disk issues.

4.8 What are the pros and cons of dongle-locked keys?

Pros:

1. You can bring them in your pocket.
2. You can pass them to other licensed users.
3. They are completely immune to machine crashes.

Cons:

1. They are comparatively slow: communication is via USB 2.0.
2. They are slightly more expensive, as the dongles are sent *FOB Origin, Freight Prepaid, & Charged Back*.

4.9 What are detachable keys?

Detachable keys can be *detached* from a pool of available keys for a specified time (up to 9999 days). Once detached, the license is automatically installed on the individual user's local machine: no further connection with the license manager is required. This can easily support commuters or people on external duties where connection with the license managers is cumbersome or inadvisable. Detaching a license requires a very small amount of network traffic (10–15 kB) and just a few seconds.

The validity of a detached license can be extended before expiration. The detached license can also be returned earlier to the license server. If not explicitly returned, the license expires on the local machine after the specified time has passed, and automatically re-materializes on the license server.

4.10 What are the pros and cons of detachable keys?

Pros:

1. They can be easily shared among users, even if they are geographically distributed in the same region (APAC, EMEA, AMER).
2. They are fast: after detach, all communications occur within the same machine.
3. A crash of the recipient machine does not cause loss of the key.

Cons:

1. If the machine hosting the license server crashes badly, the entire pool can be lost. Not a problem if the server is properly maintained and monitored.

4.11 Who chooses the license enforcing mechanism(s)?

The customer does, taking into account that:

- one node-locked key corresponds to one user;
- one dongle-locked key corresponds to two users;
- one detachable key corresponds to three users.

Suppose the customer wants to serve 8 users; this requirement can be satisfied in all ways indicated in the following table:

detachable	dongle-locked	node-locked
0	0	8
0	1	6
0	2	4
0	3	2
0	4	0
1	0	5
1	1	3
1	2	1
2	0	2
2	1	0

4.12 How many pools of detachable licenses should I have, and how large?

The factors in the decision about the number and size of each pool of detachable licenses are the following:

- A detachable license key belongs to one and only one pool.
- A pool can contain any number of license keys: such a number is fixed once the pool has been created and cannot be changed later.
- A pool can be rehosted, along with all the detachable license keys it contains, to a new server.
- A machine used as license server can host any number of pools.
- Any machine that is suitable for running ECLAIR may act as a license server, but only reliable machines should be chosen as license servers.

Given that pools cannot grow, having more than one pool hosted by the same license server is sometimes unavoidable (e.g., when new detachable licenses are to be served by the same machine). However, in case of rehosting (which must be done at the slightest sign of instability of the machine) the work to be done is multiplied by the numbers of pools. So, at the outset, it is best to plan for multiple pools only in the case of multiple license servers, each server hosting one pool.

Multiple license servers allow for the mitigation of failures: if one license server is momentarily offline, another license server may be available. This should not be overdone: if you spread your detachable keys across too many servers, users may incur overhead in finding a server with an available detachable license. A rule of thumb you can follow for each LAN is the following: if a very reliable and continuously monitored server (e.g., with a RAID disk array and S.M.A.R.T. services) is available, host a pool containing all licenses on that server. Otherwise choose two reasonably reliable machines and divide your detachable license keys into two pools, one for each license server.

Of course, if you have multiple LANs and/or multiple regions, you should first divide your detachable keys by region, then by LAN, then apply the above reasoning for each individual LAN.

4.13 What are the ECLAIR licensing requirements for Jenkins?

Each Jenkins controller requires a node-locked (ordinary) key: this key may be shared with one user working locally on the same machine.

Each Jenkins agent requires a node-locked key, which may be an ordinary one or an *analysis node key*. In the former case, the ordinary key may be shared with one user working locally on the same machine and/or with a Jenkins controller.

Analysys node keys only allow for the analysis execution and not for the generation of reports (report generation takes place in the Jenkins controller). The cost of each analysis node key contributes to the total cost as the addition of 0.5 users to the userbase.

Examples:

1. Partial coverage, userbase is n, one user works directly on the Jenkins controller, which is equipped with one ordinary node-locked license key, and all Jenkins agents, if any, are on the same machine: no extra keys are required, hence no extra cost for Jenkins.
2. As above, but with Jenkins agents all running in user machines equipped with node-locked keys: no extra keys are required, hence no extra cost for Jenkins.
3. Partial coverage, userbase is n, and there are m machines running Jenkins controllers and s machines running Jenkins agents that cannot be directly used by any of the n licensed users (for

instance, because they are remote and/or they have no display): the cost is as if the userbase was *n* + *m* + *s/2*.

As the node-locked keys for Jenkins controllers and agents are treated as increments to the userbase, the same volume discount policy applies.

Note that, when the Jenkins controller is equipped with a site/multisite/enterprise license key, users can browse the detailed reports using any of the supported web browsers even from computers without ECLAIR installed. In contrast, when the Jenkins controller is equipped with a partial license key, users can only browse the detailed reports from machines with an ECLAIR installation and a valid license key. This restriction for partial licenses only concerns detailed reports: all Jenkins users can freely browse the Jenkins' pages showing the number of ECLAIR reports, their evolution over time, and so on.

4.14 How do I get rid of expired trial licenses?

Under Windows, go to the `%CommonProgramFiles(x86)%\SafeNet Sentinel\Sentinel LDK\installed\113938\` folder. There you will find one or more files corresponding to the trial licenses you want to get rid of: their names follow a `<key id>_provisional.v2c` pattern. In order to hide them, you can remove the above mentioned files or move them elsewhere (e.g., under a `hidden` subfolder you may create for that purpose). Then you should restart the license manager, either by rebooting the machine or by doing the following from an admin prompt:

```
net stop hasplms
net start hasplms
```

You can of course do everything from the command line, e.g.:

```
cd "%CommonProgramFiles(x86)%\SafeNet Sentinel\Sentinel LDK\installed"
cd 113938
mkdir hidden
move *provisional* hidden\
net stop hasplms
net start hasplms
```

Under Linux, do the following from a superuser shell:

```
cd /var/hasplm/installed/113938/
mkdir hidden
mv *provisional* hidden/
service aksusbd stop
service aksusbd start
```

5 Maintenance

Maintenance of an ECLAIR license is the provision of the following services:

- Access to the issue-tracking system for each licensed user.
- Delivery of solutions and/or workarounds for the issues reported via the issue-tracking system.
- Delivery of updated, improved minor versions of the licensed ECLAIR packages.

5.1 Is maintenance included in the ECLAIR license?

It depends on your ECLAIR license term:

- For indefinite-term licenses, maintenance is included for the first year. Maintenance is an optional service for subsequent years: our customer service will get in touch one month before maintenance expiration with an offer for one-year maintenance renewal.
- For subscription-based licenses, maintenance is included. Our customer service will get in touch one month before license expiry with an offer for license renewal.

5.2 Am I compelled to renew maintenance?

Absolutely not: the maintenance service is optional for indeterminate licenses.

5.3 Can I keep using the issue-tracking system after maintenance expiration?

No, all the accounts connected to the maintenance contract will be disabled (not deleted).

5.4 Can I reinstate maintenance after it has expired?

Yes, reinstatement of maintenance for ECLAIR licenses is possible, unless your license refers to an ECLAIR version that reached end-of-life.

The cost is the sum of all maintenance fees since last maintenance contract expiration. Write to sales@bugseng.com for a formal quotation.

Upon reinstatement of the maintenance contract, all accounts connected to the contract that were disabled will be re-enabled.

5.5 Why have I to pay, for reinstatement, the maintenance fees of past years?

Because the work on maintaining and improving ECLAIR did continue while you were away.

6 ECLAIR Reports

ECLAIR can produce different kind of reports in different formats.

6.1 What are the main categories for ECLAIR outputs?

They are:

Detailed reports: These reports contain full details about each reported program condition (such as a coding rule violation or a possible run-time error). Here, *full details* means that all the information required for a proper understanding of the issue is presented, e.g.:

- the exact position in the source code;
- in case macros are involved, the source code both before *and* after preprocessing, and the definitions of the macros involved;

- a path in the program leading to the reported run-time error.

Summary reports: These reports contain a summary of ECLAIR findings that is suitable for the communication to third parties in order, e.g., to substantiate a claim of MISRA compliance.

Metric reports: These reports contain the values of the metrics collected for each file, function and project.

6.2 What are the main output formats?

Pure text, HTML, XML, Excel and *Open Document Format for Office Applications* (ODF, also known as *OpenDocument*). In addition, there is an internal format used to transmit detailed reports to the ECLAIR browser, which is part of the ECLAIR GUI, even though it can be used in isolation: for simplicity, we will call this format *GUI* in the sequel.

6.3 Which kinds and format of reports are available depending on the license?

The dependence is on the license coverage (see *What are the parameters defining ECLAIR license models?*). Taking into account the most used combinations, the answer is contained in the following table, where *any* denotes any license coverage whereas *SME* denotes site/multisite/enterprise coverage:

Kind/Format	Detailed	Summary	Metric
Pure text	SME	any	
HTML	SME	any	
XML	SME	SME	SME
Excel			any
ODF		any	any
GUI	any		

6.4 Why clicking on a ☞ symbol in the manual results in an error?

Most likely you overlooked the section of the manual titled "*Enabling Links from the ECLAIR User's Manual to PDF Documents*".

6.5 Is it possible to generate summaries in .docx format instead of .odt?

The ODF format has been chosen because it is a truly open format defined by an international standard (ISO/IEC 26300) that is supported by many tools. In particular, OpenDocument is supported by various versions of Microsoft Office for Windows, namely:

- Microsoft Office 2003 and Office XP (with the Open Source OpenXML/ODF Translator Add-in for Office);
- Microsoft Office 2007 (with Service Pack 2 or 3);
- Microsoft Office 2010, 2013, 2016 and 2019.

It is also supported by *LibreOffice*, a free and open-source office suite available for most popular operating systems: see https://www.libreoffice.org for more details.

Using Microsoft Office or LibreOffice you can open the `.odt` files generated by ECLAIR and save them in `.docx` format, if you wish. Visit https://bugseng.com/eclair/reports to see a summary report produces this way.

6.6 Is it possible to generate summaries in PDF?

Yes, using Microsoft Office or LibreOffice you can open the `.odt` files generated by ECLAIR and save them in PDF format. See "*Is it possible to generate summaries in .docx format instead of .odt?*" for more details.

6.7 Should I pay attention to reports of the *error* kind?

Definitely! But we have to distinguish different cases:

1. You get reports from the `B.PARSER` or the `B.TOOLCHAIN` services of ECLAIR and the compiler is failing as well (either because of parse errors or because of illegal compilation options). This splits into two further sub-cases:

   a. Failures come from a build phase where the build procedure is testing properties of the toolchain by trial and error. This happens, notably, in the standard build procedure of the Linux kernel. E.g., the build procedure may attempt compiling an empty source file with option `-march=cannonlake` to check whether such option is supported by the compiler. If it does not, the compiler will fail, and ECLAIR, which accurately matches compiler behavior, will generate a `B.TOOLCHAIN` report specifying that the retrieval of information from the toolchain has failed. Or the build procedure might, e.g., attempt compilation of

   ```
   int a[(((char)-1) < 0) ? 1 : -1];
   ```

   to check whether plain `char` is signed or unsigned. In the latter case, compilation will fail, and ECLAIR parsing will (correctly!) fail as well and generate a `B.PARSER` report. This phenomena are thus a byproduct of speculative toolchain use and do not harm correctness of the ECLAIR analysis in any way. You can prevent them by executing the speculative, self-configuring phase of the build outside the ECLAIR environment, only executing the actual build within the ECLAIR environment.

   b. Failures come from the actual build, e.g., an header file is not found, or a typo resulted in invalid syntax. The best course of action is to fix the build itself before returning to analysis with ECLAIR. Notice that you may be in this case without having noticed: some build procedures (e.g., some incorporated into popular IDEs) go on with compilation even in the case of parse errors and you may easily miss the fact that your project is not compiling cleanly. (As an aside, any build procedure not stopping at the first toolchain error is very dangerous and ought to be fixed.)

2. You get reports from the `B.PARSER` service of ECLAIR but the the compiler is not failing. Again, there are two sub-cases:

   a. You are using one of the language extensions supported by some compilers that are in sharp contrast with the applicable ISO language standards (e.g., involving a `short long` data type). These cannot be supported by ECLAIR and you would be much better off (as recommended by *all* serious coding standards) by not using such extensions.

   b. You have found a defect in ECLAIR: please file an issue on BUGSENG's issue-tracking system describing precisely your observations and allowing us to reproduce them.

7 Problems

If you run into problems, the first thing to be done after checking the questions collected in this section is to consult Chapter *Troubleshooting* of the *ECLAIR User's Manual*. If this does not allow you to solve the problem, then collect all information required to formulate a diagnosis, file an issue on BUGSENG's issue-tracking system describing precisely what is being done, what was the expected outcome, and what happened instead, making sure you attach all the relevant files (analysis scripts, ECLAIR configuration files, ECLAIR diagnostic output, and the applicable log files).

7.1 Need I Use the Latest Version of Sentinel LDK Run-Time Environment?

Yes, in order to get the latest reliability and security fixes. Typically each minor or major ECLAIR version (see *What is the meaning of ECLAIR version numbers?*) installs the *Sentinel LDK Run-Time Environment* (RTE) version that was current when the ECLAIR release was closed.

Point your browser to http://127.0.0.1:1947/_int_/about.html in order to identify the current RTE version. If the above page does not display, it means the RTE is not installed, or it is not running, or it is not reachable. Common causes are:

- You skipped installation of the *Sentinel LDK RTE* component in the ECLAIR installer.
- You attempted installation without administration rights.
- You forgot to disable antivirus/antimalware software during the installation.
- Sentinel LDK communicates via TCP and UDP on port 1947. This port is IANA-registered exclusively for this purpose. The firewall must be configured so that communication via this port is not blocked.

The easiest way to install a recent version of Sentinel LDK RTE is to install the latest version of ECLAIR. If your maintenance has expired (see *Maintenance*), you can still upgrade Sentinel LDK RTE by following the instructions in KB0022151[5].

7.2 I get a Sentinel LDK Protection System error code of the form Hnnnn: why?

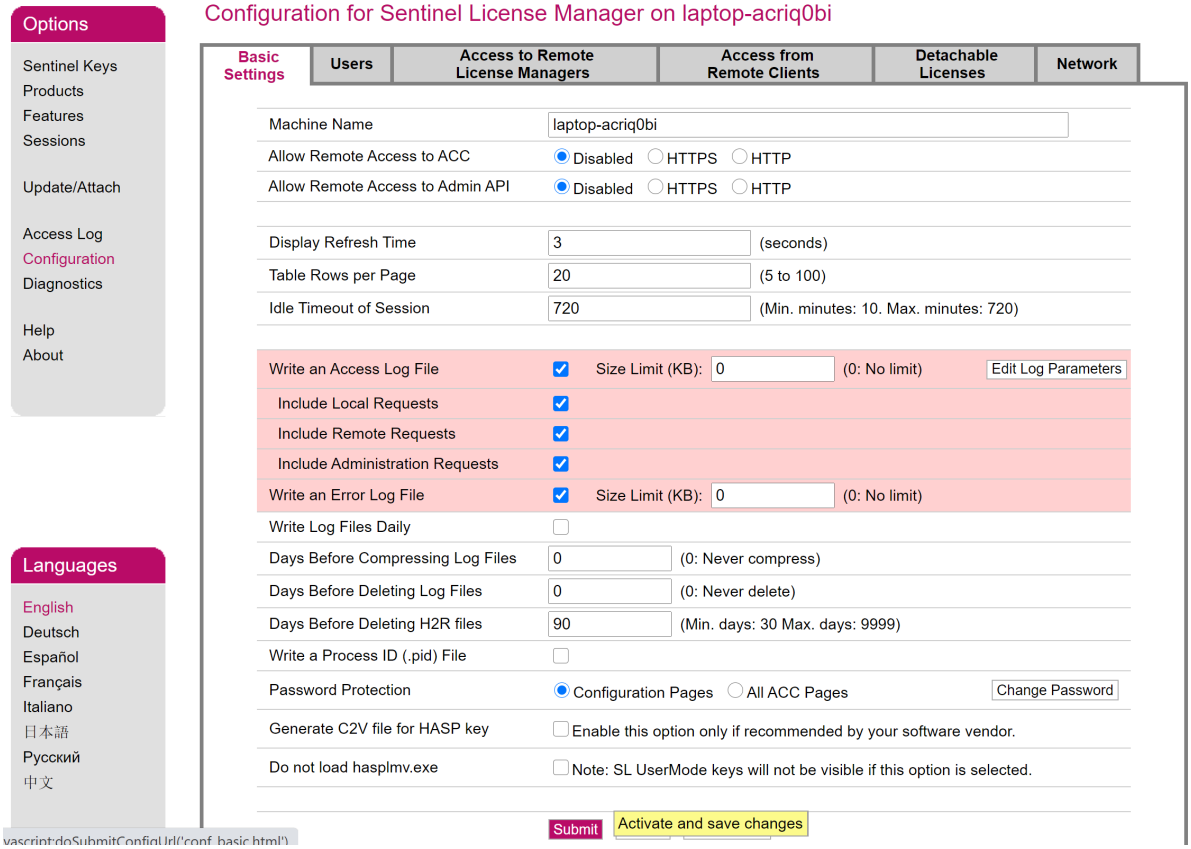
If you get an error from the *Sentinel LDK Protection System* like `Sentinel key not found (H0007)` or `Terminal services detected (H0027)` or `Unable to access Sentinel Run-time environment (H0033)` or `Feature has expired (H0041)`, or anything with an `Hnnnn` code, where `nnnn` is a four-digit number, please make the following checks:

1. Did you install or upgrade *Sentinel LDK RTE* as recommended by the installation procedure? If not, please do that.
2. If you are using a trial license, is the trial period expired? You can check this as explained in the *ECLAIR Evaluation Guide*.
3. If you are not using a trial license, did you activate your license by applying the activation file provided by BUGSENG? Check the applicable *ECLAIR Quick Installation Guide* if you are unsure.
4. Are you getting error `Terminal services detected (H0027)`? If you have a node-locked license key, this is normal: remote access is not allowed (see "*What are the available license-enforcing mechanisms?*" for more details).

[5] https://supportportal.thalesgroup.com/csm?id=kb_article_view&sys_kb_id=f0ffac121bd11850ce59fcc1cd4bcb34

5. Are you getting error `Feature has expired (H0041)` but you have bought an indeterminate license? This may be because you have previously had a trial license installed, and this error is hiding the real issue. Follow the instructions in "*How do I get rid of expired trial licenses?*" and then retry.

If following the above steps does not solve the problem, please activate logging in *Sentinel Admin Control Center* (ACC), which is accessible using any web browser at URL http://localhost:1947. Use the **Configuration** option of the menu on the left-hand side and make sure the boxes for **Write an Access Log File**, **Include Local Requests**, **Include Remote Requests**, **Include Administration Requests** and **Write an Error Log File** are all checked, then click on the **Submit** button.



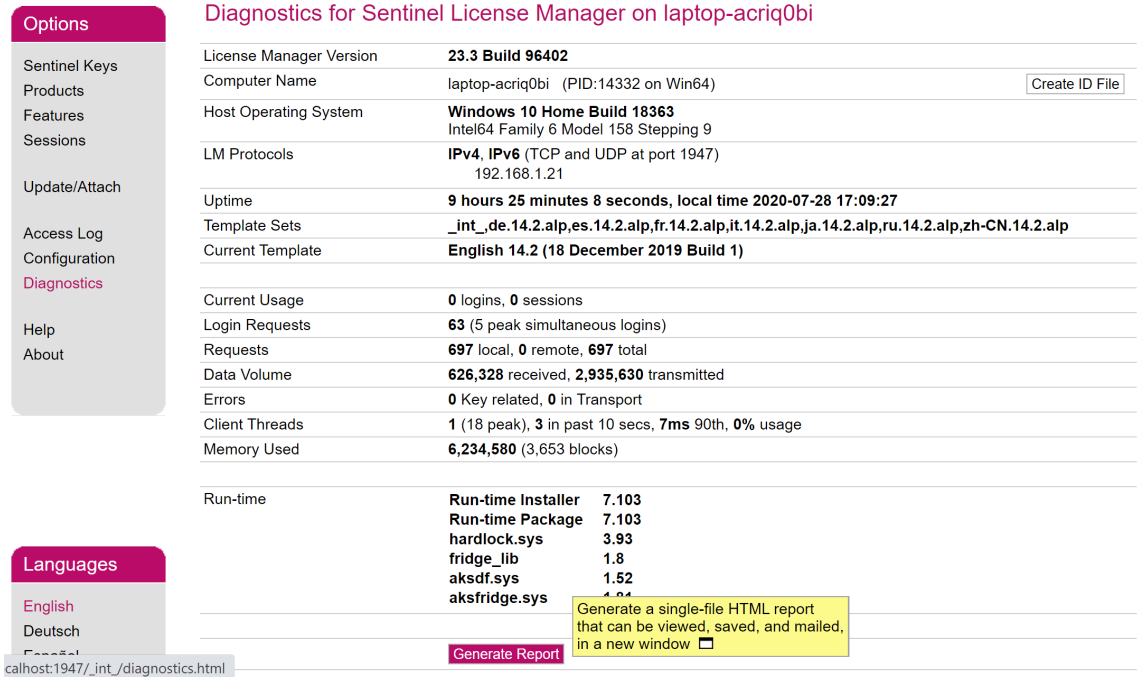
To enable extra logging, create a file named `hasp_113938.ini` containing the following two lines:

```
requestlog=1
errorlog=1
```

If on Windows, copy this file under the `%LocalAppData%\SafeNet Sentinel\Sentinel LDK\` folder; if on Linux, copy this file under `$HOME/.hasplm` directory.

Then reproduce the error and save files `access.log` and `error.log` (if present) that are created under the `%CommonProgramFiles(x86)%\Aladdin Shared\HASP\` folder, in Windows, and under the `/var/hasplm` directory, in Linux. Save also the files `access_113938.log` and `error_113938.log` (if present) that are created under the `%LocalAppData%\SafeNet Sentinel\Sentinel LDK\` folder, in Windows, and under the `$HOME/.hasplm` directory, in Linux.

In order to provide us with all data we need to give you assistance, use also the **Diagnostics** option of Sentinel ACC and click on the **Generate Report** button; then save the web page containing the report as `diagnostics.html`.



Finally, open a *new* issue on BUGSENG's issue-tracking system, describing the problem you are observing and attaching all files mentioned above.

7.3 Running the analysis in the GUI fails, but it succeeds outside the GUI: why?

A common cause for this problem is that your setting for the `PATH` environment variable (in the **Environment** section of the GUI) omits some essential paths. For instance, under Windows the `PATH` environment variable should contain, at the very least, `C:/Windows/System32` and `C:/Windows`. In addition, you will have to add all directories your build procedure and toolchain assume to be in `PATH`. In order to make sure the contents of `PATH` is sufficient, you can open a DOS/shell prompt, set `PATH` exactly as you have set it in the GUI, and run the build procedure manually.

Please note that the fact that the GUI does no inherit `PATH` from the invoking environment is a feature: fuzzy control of `PATH` is one of the most frequent causes where the wrong build is analyzed (something that ECLAIR has been specifically designed to avoid).

7.4 A project (possibly an ECLAIR demo project) does not work: why?

One possibility is that you stumbled upon the maximum path length limitation of the Windows API[6]. A possible solution is to shorten path lengths (e.g., by moving the project up in the file system tree). Another possible solution is enabling long paths[7].

[6] https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file#maximum-path-length-limitation
[7] https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file#enable-long-paths-in-windows-10-version-1607-and-later

7.5 ECLAIR seems to be ignoring the code in my project: why?

See "*How can I make sure my toolchain is properly intercepted by ECLAIR?*"

If the toolchain is properly intercepted and you still are not seeing reports for a portion or all of your code, chances are you are in a situation where both the following conditions hold:

- you are using the `-project_root=`*`PROJECT_DIRECTORY`* option of `eclair_env`, and part or all your project sources are not below the *`PROJECT_DIRECTORY`* argument;
- you are using an `eclair_env` configuration, like

```
-reports+={hide,all_exp_external}
```

  that hides all report areas tagged `external`.

As all areas not below the argument of `-project_root` are tagged as `external`, this implies that such areas are hidden, and a report whose areas are all hidden is itself hidden.

If you are unsure where the sources of the build you are analyzing are located, the best course of action is the following:

1. Run an ECLAIR analysis using the `-project_root=/dev/null` option for `eclair_env`. You will get reports where all file paths are absolute and, clicking on *Number of analysis files* in the ECLAIR detailed report, you will obtain a listing of all files involved in the build.
2. Now you can use the *-project_root={PROJECT_DIRECTORY}* option of `eclair_env`, choosing as *`PROJECT_DIRECTORY`* the lowest directory in the file system such that all your source files and all the ECLAIR configuration files (those with the `.ecl` extension) are in or below that directory.

Regarding point 2, for most applications the header files belonging to the C/C++ implementation you are using need not be below *`PROJECT_DIRECTORY`* as they can legitimately be considered to be external to the analyzed project. Still on point 2, if you are in Windows and your sources span multiple drives, then you should keep the `-project_root=/dev/null` option and, unless you have special needs, explicitly tag as `external` the header files belonging to the C/C++ implementation.

7.6 The ECLAIR Bug Finder gives a false positive: will you fix it?

Technically speaking, the *ECLAIR Bug Finder* does not give *positives*: see the definition of "caution report" in the *Glossary* chapter of the *ECLAIR User's Manual*. However, for the sake of simplicity, let us improperly use the term "false positive" to denote also "a caution about a symptom that does not correspond to an actual problem."

The *ECLAIR Bug Finder* is a very fast static analyzer that deals with the unescapable complexity/precision tradeoff of static analysis by favoring low computational complexity (i.e., high speed of execution) at the expense of precision. As a result, the *ECLAIR Bug Finder* has both *false positives*[8] and false negatives. To be more precise, in order to have a small number of false positives, it has a significant number of false negatives. It is a rather delicate compromise: in order to maintain analysis speed, "fixing false positives" may easily result in way too many false negatives.

Please note that bug finders have little to do with MISRA compliance [2]: they allow finding some bugs (which is always nice), but MISRA compliance is a different thing. The *ECLAIR Bug Finder* does find several non-trivial bugs in very short time, but for MISRA compliance you have to use one of the

[8] In the improper sense we stipulated to attribute to the expression in this FAQ.

dedicated packages (see "*What is an ECLAIR package?*"). Note also that the *ECLAIR Bug Finder* gives much better results with programs that are nearly compliant to MISRA C/C++ or to BARR-C:2018, that is, program that refrain from using the language in ways that make data-flow and control-flow analyses (and human reasoning) more difficult.

All that is not to imply that we do not welcome reports about the *ECLAIR Bug Finder*: we keep improving all components of ECLAIR and your feedback is extremely important to us.

References

[1] R. Bagnara. That's C, baby. C! 2019. Available at https://bugseng.com/that-is-c-baby-c.

[2] R. Bagnara, A. Bagnara, and P. M. Hill. The MISRA C coding standard and its role in the development and analysis of safety- and security-critical embedded software. In A. Podelski, editor, *Static Analysis: Proceedings of the 25th International Symposium (SAS 2018)*, volume 11002 of Lecture Notes in Computer Science, 5–23. Freiburg, Germany, 2018. Springer International Publishing.

BUGSENG srl
Parco Area delle Scienze 53/A
I-43124 Parma, Italy
Via Lenin 132/F
I-56017 San Giuliano Terme, Italy
Email: info@bugseng.com
Web: http://bugseng.com