# ECLAIR Frequently Asked Questions

## Contents

This document is subject to change without notice. Last modification: Wed, 22 Jan 2025 11:06:38 +0100.

# 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:

- MC: MISRA C + Essentials[1]
- MP: MISRA C++ + Essentials[2]
- B: BARR-C:2018 + Essentials[3]

All MISRA packages include the B package. Packages can be combined in all possible ways into products so that, e.g., product MCP contains all features of MC and of MP.

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

Exactly.

## 2.4 What is the "ECLAIR *client kit*"?

The ECLAIR *client kit* is a small software component that contains all and only what is necessary to use the ECLAIR analysis results from within one of the supported IDEs (*Eclipse* and its derivatives, *IntelliJ IDEA* and its derivates, *Visual Studio Code*, *Microsoft Visual Studio*, *NetBeans* and its derivatives and *GNU Emacs*). Note that the ECLAIR *client kit* has no static analysis capability and can only be used in conjunction with one or more ECLAIR packages; on the other way around, the entire functionality of the kit is included in all ECLAIR packages. Use of the ECLAIR *client kit* is recommended in two scenarios:

1. When analysis is centralized (e.g., on one or more Jenkins installations), and users want to reduce the installed software on their PCs. In this case using the kit or a package is only a matter of user convenience and does not result in any price difference.
2. In conjunction with a *project license*, in which case the licensing for the kit is negotiated along with the project license itself (see *What are the parameters defining ECLAIR license models?*).

## 2.5 What are the basic hardware and software requirements of ECLAIR?

ECLAIR can run on physical and virtual machines running reasonably recent versions of Windows, Linux, or macOS. As static analysis is, in general, a tough job, a generous amount of computing power and RAM memory is recommended.

[1] https://www.bugseng.com/sites/default/files/resources/ECLAIR_MC.pdf

[2] https://www.bugseng.com/sites/default/files/resources/ECLAIR_MP.pdf

[3] https://www.bugseng.com/sites/default/files/resources/ECLAIR_B.pdf

## 2.6 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.

## 2.7 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.

## 2.8 Is ECLAIR suitable for safety- and security-related development?

ECLAIR was designed at the outset taking into account even the most stringent requirements of critical software development. The ECLAIR functional safety ecosystem offers multilevel solutions, supporting a gradual approach to qualification.

ECLAIR qualification by validation can be achieved with very high levels of confidence and little effort using the ECLAIR Qualification Kits[4], which are available for all common functional safety standards. An *ECLAIR Qualification Kit* contains, in addition to all documentation required to comply with the chosen functional safety standard, a large number of validated testcases and automation machinery that facilitates running ECLAIR on all of them and collecting the required evidence. For increased independence, *ECLAIR Qualification Kits* are optionally supplied with third-party testsuites.

[4] https://www.bugseng.com/sites/default/files/resources/ECLAIR_Qualification_Kits.pdf

Alternatively, for the cases when the tool operational environment can be exactly reproduced (typically by means of virtual machines or suitable containers), BUGSENG offers an ECLAIR Qualification Service[5] that spares customers from most of the burden.

In addition, ECLAIR is certified by TÜV SÜD for use in safety-critical development[6] in compliance with ISO 26262 up to ASIL D[7], IEC 61508 up to SIL 4[8], EN 50128 up to SIL 4[9], EN 50657 up to SIL 4[10], IEC 62304 up to Class C[11], ISO 19014 up to MPLr e[12], and ISO 25119 up to SRL 3[13]. The ECLAIR FuSa Pack[14], which can be licensed for single as well as for multiple projects, consists of ECLAIR's TÜV SÜD certificate and technical report, and of the *ECLAIR Safety and Security Manual*, with all information and checklists for the adoption of ECLAIR in safety-related development.

### 2.9 Can I use ECLAIR for requirements traceability?

Yes, you can. Starting from version 3.13.0, ECLAIR 3.* packages include a requirements management tool as well as a service for the automatic checking of the traceability between requirements and program entities: this is prescribed by the MISRA coding guidelines and all functional-safety standards.

### 2.10 Can ECLAIR be used to automate part of the compiler qualification process?

Yes, if you have a license of SuperTest and a license of ECLAIR you can take the most out of your tools by integrating *CerTran*. The CerTran extension uses ECLAIR's functionalities to automate the configuration of *SuperTest*[15] for compiler qualification by scanning the application build process and creating the exact test configuration files needed to cover all use cases. This can be completely incorporated into a Continuous Integration system. Not only does this save a considerable amount of time, but it also avoids configuration errors that can easily occur when scanning the build process manually. See also section 3 Toolchains. CerTran is an add-on to an ECLAIR license: it can be combined with all ECLAIR packages although it can be purchased separately. Get in touch with one of our sales representatives for a quote.

## 3 Toolchains

One of the key features of ECLAIR is that it completely automates any configuration concerning the precise C/C++ dialects used in the project: ECLAIR will detect by itself, without any user intervention, all the relevant implementation-defined behaviors, no matter whether and how compiler options are used to influence them. It does so by intercepting all the invocations to the toolchain components (compiler, assembler, librarian, linker), by interpreting the meaning of each provided options, and by "knowing" the defaults of each supported compiler.

It is precisely thanks to its ability to catch all the options given to the compiler for each translation unit that is compiled, that ECLAIR knows exactly the set of all and only the compiler use cases for your project. This proves crucial when it comes to compiler qualification. Leveraging this feature, the most manual and error-prone part of the process can be automated with the *CerTran* ECLAIR extension (see "*Can ECLAIR be used to automate part of the compiler qualification process?*" for more details).

[5] https://www.bugseng.com/sites/default/files/resources/ECLAIR_Qualification_Service.pdf

[6] https://www.bugseng.com/sites/default/files/resources/ECLAIR_TUV-SUD_Certificate.pdf

[7] https://www.bugseng.com/sites/default/files/resources/ECLAIR_ISO26262.pdf

[8] https://www.bugseng.com/sites/default/files/resources/ECLAIR_IEC61508.pdf

[9] https://www.bugseng.com/sites/default/files/resources/ECLAIR_EN50128.pdf

[10] https://www.bugseng.com/sites/default/files/resources/ECLAIR_EN50657.pdf

[11] https://www.bugseng.com/sites/default/files/resources/ECLAIR_IEC62304_FDA.pdf

[12] https://www.bugseng.com/sites/default/files/resources/ECLAIR_ISO19014.pdf

[13] https://www.bugseng.com/sites/default/files/resources/ECLAIR_ISO25119.pdf

[14] https://www.bugseng.com/sites/default/files/resources/ECLAIR_FuSa_Pack.pdf

[15] SuperTest is a renowned compiler test and validation suite developed and commercialised by Solid Sands B.V.

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

We add support for new toolchain releases on a regular basis, giving priority to the most used ones in the embedded system industry. So we ask about which toolchain(s) you are using 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®, AndeSoft™ SW Stack, CAES, clang/LLVM and its derivatives, CodeWarrior™, Cosmic Software, CrossWorks™, ESP-IDF, Emscripten, Freescale™, GCC and its derivatives, Green Hills®, HighTec, IAR™, Infineon™, Intel®, Keil Software®, MPLAB®, MPLAB®, Melexis™, Microsoft®, MinGW-w64 NXP®, QNX™, Renesas Electronics, SOFTUNE™, TASKING®, Texas Instruments™, Wind River®, Xilinx™, xPack.

## 3.3 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 and to give us a couple of weeks to implement it.

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

Look at the *Number of analysis frames* table in the ECLAIR analysis 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 analysis 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.

**License key:**

A software or hardware token that entitles a user or a machine to the use a selected ECLAIR features.

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/GitLab/GitHub controllers/instances and agents/runners (see "*What are the ECLAIR licensing requirements for Jenkins, GitHub and GitLab?*" 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 software developers and all software quality assurance people at a given site, independently from whether, at any given time, they use ECLAIR directly, they use ECLAIR reports, or they do not use ECLAIR and its outputs at all. Here "site" has to be interpreted in its broadest sense: a clearly-delimited and official company department, division or group may qualify as a site. External consultants temporarily working on software development or quality assurance have to be counted.

**Multisite:**

Same as above, but for a given set of sites, provided 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 are the advantages of site/multisite/enterprise licenses?

Such licenses allow the generation of *detailed outputs* in all supported format for their distribution within the site/multisite/enterprise. For instance, HTML outputs 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 outputs* can only be consulted using the ECLAIR analysis results browser. *Rich outputs*, *summary outputs* and *metric outputs* can be generated in any supported format with any license (see "*What are the main categories for ECLAIR outputs?*" for an explanation of the different output categories).

The main advantage of site/multisite/enterprise licenses is that each human user can obtain, at the customer's discretion, any kind of license key, thereby relaxing the constraints specified in "*Who chooses the license enforcing mechanism(s)?*"

## 4.6 What are the available license-enforcing mechanisms?

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

**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.

**Detachable:**
The key is fully implemented in software and can be detached by a (possibly remote) license server.

For partial coverage licenses, one detachable key serves three users in the same region (APAC, EMEA, AMER).

The above distinction concerns the technology used to implement keys. An orthogonal distinction is the one between *ordinary keys* and *analysis node keys*: the latter are only meaningful in the context of ECLAIR deployment in continuous integration system (see "*What are the ECLAIR licensing requirements for Jenkins, GitHub and GitLab?*").

## 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 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 key requires a very small amount of network traffic (10–15 kB) and just a few seconds.

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

## 4.9 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.

## 4.10 Who chooses the license enforcing mechanism(s)?

The customer does, taking into account that:

- one node-locked key corresponds to one user;
- one detachable key corresponds to three users in the same region (APAC, EMEA, AMER) for partial coverage licenses, but only to one user for site/multisite/enterprise licenses.

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

| detachable | node-locked |
|---|---|
| 0 | 8 |
| 1 | 5 |
| 2 | 2 |

A customer with a site/multisite/enterprise license can serve all its users with any combination of keys, with at most one key per user.

## 4.11 How many pools of detachable license keys should I have, and how large?

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

- A detachable license key belongs to one and only one pool and cannot be transferred to a different pool.
- A pool can contain any number of license keys: such a number can be increased at any time, but it cannot be decreased.
- 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.

Having more than one pool hosted by the same license server is only advisable as a temporary measure, e.g., because we know some pools will have to be rehosted to a different machine. 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 key. 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 license keys 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.

## 4.12 How do I work with detachable license keys?

For human users, everything can be conveniently done via *Sentinel Admin Control Center* (Sentinel ACC), which you can access by pointing your browser to http://localhost:1947/ See this video[16] for a tutorial.

If you want to automate the detaching and canceling operations, the **`eclair_licman`** program is what you need.

[16] https://www.youtube.com/watch?v=8ApXrqHsyMk

## 4.13 ECLAIR works even if I do not detach the key? So what?

It works, but take into account that the analysis with ECLAIR of a typical project involves thousands of accesses to the key, so:

- depending on the latency and speed of the network connection between your machine and the license server, the analysis may be significantly slowed down;
- if you do not detach the key, you are subject to race conditions: suppose your analysis requires an hour and 1000 accesses to the key, and that the very last access fails because your colleagues are using all the available keys.

The only sensible case where you do not want to detach the key is when ECLAIR runs in a Docker container and the license server is running on the Docker host.

## 4.14 Should I take any precautions concerning my license keys?

Sure, you need not lose them: there are hard costs associated with them, so they cannot be replaced for free and replacement requires paperwork as well.

**For node-locked keys and pools of detachable keys:**

1. Never dispose or reformat or change disks of a machine unless you have already rehosted all node-locked keys and pools of detachable keys hosted by that machine.
2. Especially for large pools of detachable keys, use RAID disk arrays and S.M.A.R.T. services (in general, such large pools should be hosted by machines continuously monitored by competent IT personnel).

## 4.15 What are the ECLAIR licensing requirements for Jenkins, GitHub and GitLab?

Each Jenkins controller and GitHub/GitLab instance requires an ordinary key: this key may be shared with one user working locally on the same machine.

Each Jenkins agent and GitHub/GitLab runner requires a 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/GitHub/GitLab controller/instance.

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

Examples with Jenkins, equally applicable to GitHub and GitLab:

1. Partial coverage, userbase is $n$, one user works directly on the Jenkins controller, which is equipped with one ordinary 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 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 cardinality was $n + m + s/2$.

As the keys for Jenkins/GitHub/GitLab controllers/instances and agents/runners are treated as increments to the userbase, the same volume discount policy applies.

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

### 4.16 How do I get rid of expired trial license keys?

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 license keys 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 indefinite-term 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 with maintenance reinstatement you receive all software updates and improvements implemented while your maintenance was on hold.

## 6 ECLAIR Outputs

ECLAIR can produce different kind of outputs in different formats.

### 6.1 What are the main categories for ECLAIR outputs?

They are:

**Summary outputs:**
These outputs contain comprehensive information about the analysis as well as counts of the issues uncovered by the analysis per service, per file, per tag and combinations thereof.

**Rich outputs:**
These outputs contain comprehensive information about the analysis results without going into the detail of each reported program condition. For each individual issue reported by the analysis, these outputs contain:

- the name of the file where the first code area resides;
- if the report has been tagged (e.g., to capture a MISRA deviation), the tag, along with any documentation associated to the application of the tag to the report (e.g., the justification for a MISRA deviation).

These outputs are suitable for the communication to third parties in order, e.g., to precisely substantiate a claim of MISRA compliance.

**Detailed outputs:**
These outputs contain comprehensive information about the analysis results, including all details about each reported program condition (such as a coding rule violation or a possible run-time

error). In other words, these output contain all the information required for a proper understanding of each individual issue reported by the analysis, e.g.:

- the exact position in the source of all the involved code areas;
- messages indicating precisely which role each code area plays in the reported program condition;
- 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 issue;
- if the report has been tagged (e.g., to capture a MISRA deviation), the tag, along with any documentation associated to the application of the tag to the report (e.g., the justification for a MISRA deviation).

Such level of detail, while essential for whoever is in charge of acting upon undesirable program conditions, is undesirable for almost any other application. Moreover, such level of detail can effectively be navigated only via a browser. For this reason, when detailed outputs have to be printed, the level of detail can be decreased. The minimum level of detail for a detailed output is the line number and the message for the first area in the report.

**Metric outputs:**
These outputs 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, Word, Excel, LibreOffice Writer, and LibreOffice Calc. In addition, there is an internal protocol used to transmit detailed outputs to the ECLAIR browser.

## 6.3 Which kinds and format of outputs 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 | Rich | Summary | Metric |
|---|---|---|---|---|
| **Pure text** | SME | any | any | any |
| **HTML** | SME | any | any | |
| **XML** | SME | any | any | |
| **Word** | SME | any | any | |
| **Excel** | SME | any | any | any |
| **LibreOffice Writer** | SME | any | any | |
| **LibreOffice Calc** | SME | any | any | any |
| **Browser** | any | any | any | 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 outputs in PDF?

Yes. For example, Word, Excel and LibreOffice allow saving documents in PDF format.

### 6.6 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. These 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.

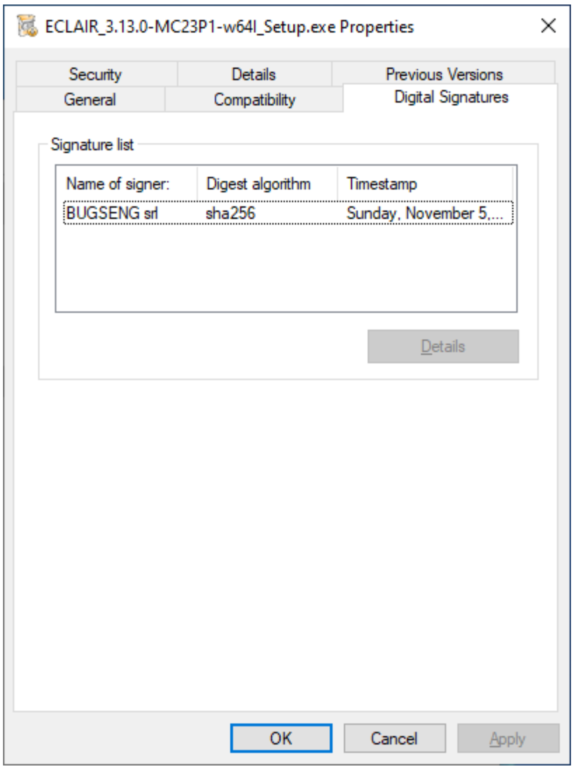
## 7 Problems

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 Why is the publisher of the ECLAIR installer unknown?

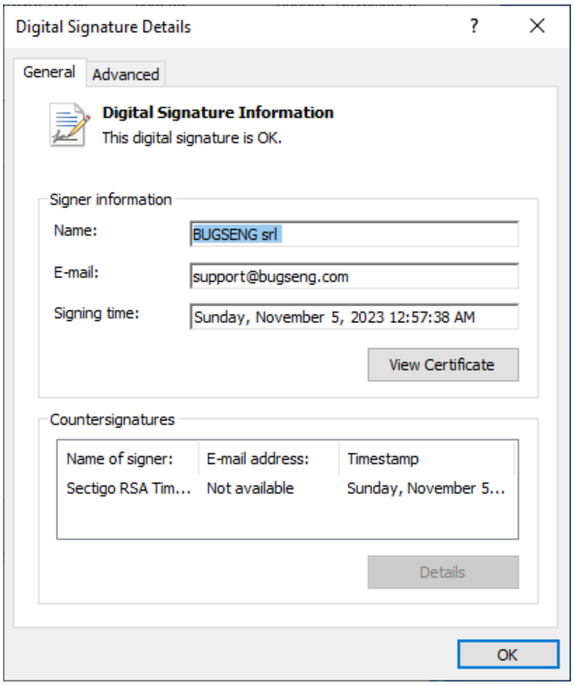
Sometimes, Windows does not seem to automatically find the digital signature of the installer.

To help the operating system in finding the product signature, perform the following steps. Right click on the installer, click on **Properties** and select the **Digital Signatures** tab which looks like the following:



| Name of signer: | Digest algorithm | Timestamp |
|---|---|---|
| BUGSENG srl | sha256 | Sunday, November 5,... |

Select the item shown in the signature list having BUGSENG srl as signer. This will enable the **Details** button, click it and a window showing information about the digital signature will appear:



Digital Signature Details

General | Advanced

**Digital Signature Information**

This digital signature is OK.

Signer information

Name: BUGSENG srl

E-mail: support@bugseng.com

Signing time: Sunday, November 5, 2023 12:57:38 AM

Countersignatures

| Name of signer: | E-mail address: | Timestamp |
|---|---|---|
| Sectigo RSA Tim... | Not available | Sunday, November 5... |

Click the **OK** button to close the window. Now, lauch the ECLAIR installer and BUGSENG srl will be correctly recognized as the publisher.

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

Yes, for a number of reasons, including 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. Starting from version 3.13.0 installation of Sentinel LDK Run-Time Environment (RTE) is no longer optional.

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[17].

**Warning**

You must *never* downgrade Sentinel LDK RTE!

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

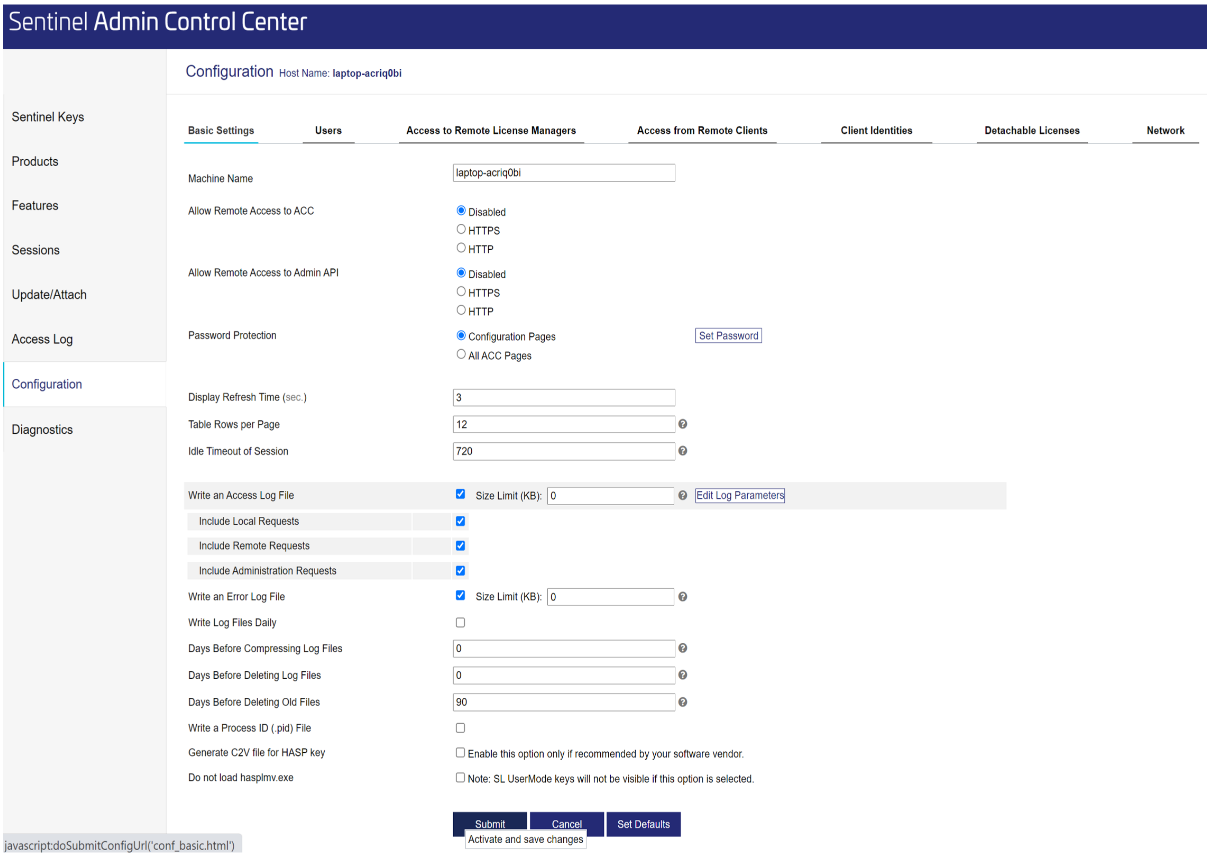
Doing so you risk breaking your ECLAIR installation. Most importantly, you risk losing *all* your Sentinel keys in a non-recoverable way. If you find yourself in need of downgrading your ECLAIR installation, please make sure you answer *no* to the installer question whether you want to install a version of Sentinel LDK RTE that is possibly older than the one you already have on the system.

## 7.3 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. Are you getting error `Feature has expired (H0041)` but you have bought an indefinite-term 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 license keys?*" 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.



| | |
|---|---|
| Machine Name | laptop-acriq0bi |
| Allow Remote Access to ACC | ● Disabled<br>○ HTTPS<br>○ HTTP |
| Allow Remote Access to Admin API | ● Disabled<br>○ HTTPS<br>○ HTTP |
| Password Protection | ● Configuration Pages<br>○ All ACC Pages |
| Display Refresh Time (sec.) | 3 |
| Table Rows per Page | 12 |
| Idle Timeout of Session | 720 |
| Write an Access Log File | ☑ Size Limit (KB): 0 |
| Include Local Requests | ☑ |
| Include Remote Requests | ☑ |
| Include Administration Requests | ☑ |
| Write an Error Log File | ☑ Size Limit (KB): 0 |
| Write Log Files Daily | □ |
| Days Before Compressing Log Files | 0 |
| Days Before Deleting Log Files | 0 |
| Days Before Deleting Old Files | 90 |
| Write a Process ID (.pid) File | □ |
| Generate C2V file for HASP key | □ Enable this option only if recommended by your software vendor. |
| Do not load hasplmv.exe | □ Note: SL UserMode keys will not be visible if this option is selected. |

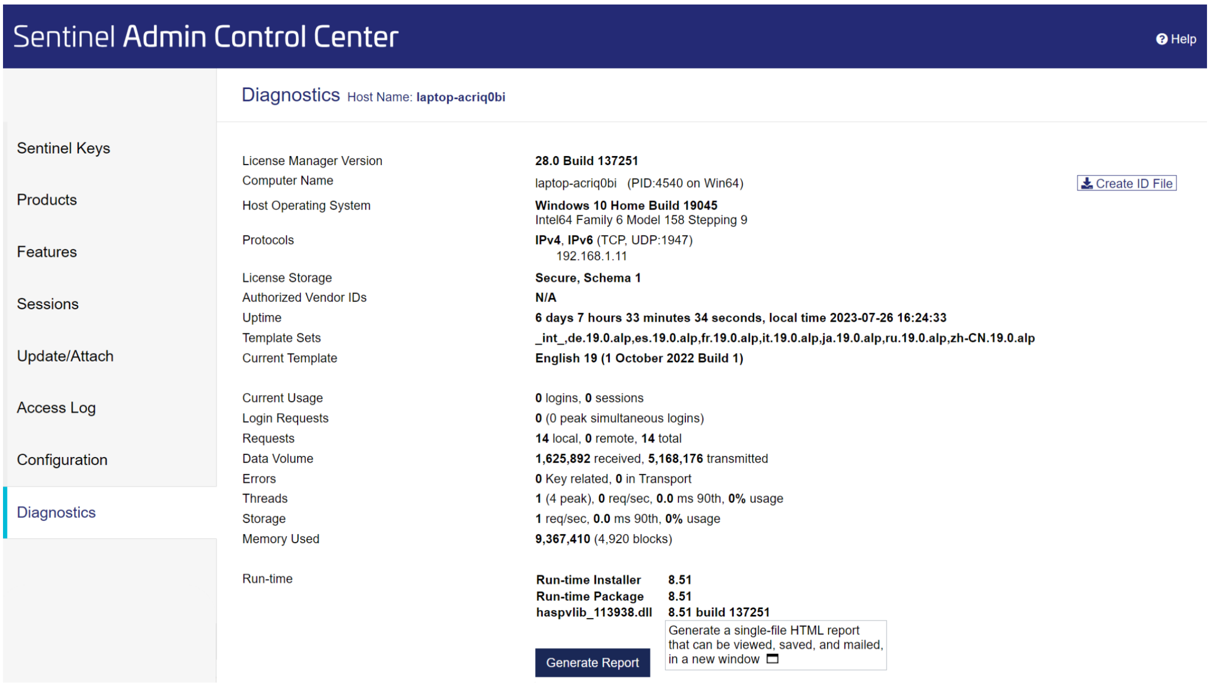
To enable extra logging, create a file named `hasp_113938.ini` containing the following two lines:

```
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`.



Diagnostics Host Name: **laptop-acriq0bi**

| | |
|---|---|
| License Manager Version | **28.0 Build 137251** |
| Computer Name | laptop-acriq0bi (PID:4540 on Win64) |
| Host Operating System | **Windows 10 Home Build 19045**<br>Intel64 Family 6 Model 158 Stepping 9 |
| Protocols | **IPv4**, **IPv6** (TCP, UDP:1947)<br>192.168.1.11 |
| License Storage | **Secure, Schema 1** |
| Authorized Vendor IDs | **N/A** |
| Uptime | **6 days 7 hours 33 minutes 34 seconds, local time 2023-07-26 16:24:33** |
| Template Sets | **_int_,de.19.0.alp,es.19.0.alp,fr.19.0.alp,it.19.0.alp,ja.19.0.alp,ru.19.0.alp,zh-CN.19.0.alp** |
| Current Template | **English 19 (1 October 2022 Build 1)** |
| Current Usage | **0** logins, **0** sessions |
| Login Requests | **0** (0 peak simultaneous logins) |
| Requests | **14** local, **0** remote, **14** total |
| Data Volume | **1,625,892** received, **5,168,176** transmitted |
| Errors | **0** Key related, **0** in Transport |
| Threads | **1** (4 peak), **0** req/sec, **0.0** ms 90th, **0%** usage |
| Storage | **1** req/sec, **0.0** ms 90th, **0%** usage |
| Memory Used | **9,367,410** (4,920 blocks) |
| Run-time | **Run-time Installer** **8.51**<br>**Run-time Package** **8.51**<br>**haspvlib_113938.dll** **8.51 build 137251** |

Generate a single-file HTML report that can be viewed, saved, and mailed, in a new window

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

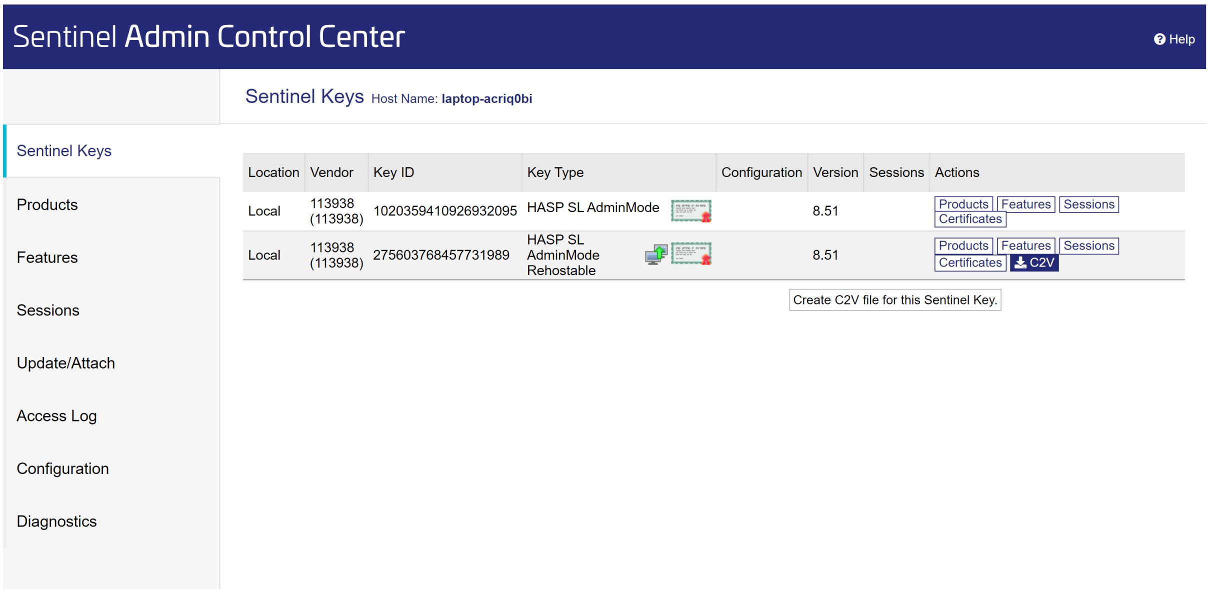
## 7.4 My Sentinel key is marked as *Cloned* and all features *Disabled*: why?

Sentinel LDK keys are protected against cloning for obvious reasons.

On physical machines, Sentinel LDK uses various components such as CPU, ethernet card, optical drive, PCI card slot peripherals (for example: display, storage, network, multimedia) along with the hard drive serial number and motherboard ID to verify fingerprints. The identities of these physical devices will be checked against reference identities: when the differences exceed a certain threshold, the key is marked as cloned. The threshold is there to ensure that, e.g., simply replacing one optical drive does not trigger disabling the key.

On virtual machines, Sentinel LDK checks: the virtual MAC address, the available CPU characteristics, and the UUID (*Universal Unique IDentifier*) of the virtual machine. In addition, Sentinel LDK prevents attacks (against a protected application like ECLAIR) that are based on virtual machine rollback snapshots by detecting that a time shift event may have occurred.

If you are affected by this problem, please open a new issue on BUGSENG's issue-tracking system, explain what happened and attach the *C2V file* of the affected key(s). You can obtain that file simply by clicking on the C2V button at the right of the key:



Sentinel Keys Host Name: **laptop-acriq0bi**

| Location | Vendor | Key ID | Key Type | Configuration | Version | Sessions | Actions |
|---|---|---|---|---|---|---|---|
| Local | 113938<br>(113938) | 1020359410926932095 | HASP SL AdminMode |  | 8.51 |  | Products Features Sessions<br>Certificates |
| Local | 113938<br>(113938) | 275603768457731989 | HASP SL<br>AdminMode<br>Rehostable |  | 8.51 |  | Products Features Sessions<br>Certificates C2V |

Create C2V file for this Sentinel Key.

## 7.5 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).

## 7.6 Can ECLAIR analyze a project located in a Virtual Box shared folder?

No, it cannot. VirtualBox shared folders are implemented with a very minimal file system, that just provides basic features to read/write files from/to the host machine. Many applications, including ECLAIR, need advanced features like file locking and access controls, which are not implemented for shared folders. Analysis of projects located in such folders must be avoided.

## 7.7 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[18]. 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[19].

## 7.8 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;

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

- 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 analysis 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.

## 7.9 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*[20] 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 [1]: 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 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.

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

### 7.10 An ECLAIR IDE plugin is not able to connect to the host: why?

When using the IDE plugins by specifying the `Eclair Report Host` with a DNS alias, it may happen that the system is unable to resolve the alias. In this case the plugin reports the following error: `FATAL: Failed to connect to server eclair_report: host not found`. Provide the IP address directly in order to avoid the error.

### 7.11 The ECLAIR GUI does not work: why?

The ECLAIR GUI for Linux requires fairly recent versions of `libgbm` that are not always available from the package managers of some legacy operating systems. All operating systems supported by ECLAIR come with a suitable version of the library already installed. Some unsupported systems may not provide the library, in which case the following error will be produced by the `eclair_gui` command: `./eclair_gui: symbol lookup error: ./eclair_gui: undefined symbol: gbm_bo_get_modifier`. A possible fix for that problem is to manually copy the `libgbm.so.1` file taken from Ubuntu 18.10 to the `<ECLAIR_INST_DIR>/bin/gui` directory. The fix is not guaranteed to work in all cases. Furthermore, using ECLAIR on unsupported systems is not recommended.

### 7.12 Is there a list of known ECLAIR issues?

Of course: this is available to all users with a valid maintenance contract in the *Known issues* project of BUGSENG's issue-tracking system. Known issues are organized under "umbrella issues" each corresponding to a version of ECLAIR. For instance, all ECLAIR 3.11.0 issues are children of the umbrella issue whose summary contains "All ECLAIR 3.11.0 known issues". Users of a certain version of ECLAIR are strongly invited to monitor the corresponding umbrella issue by clicking on the *Monitor* button in the lower left corner of the *View Issue Details* pane: this way, they will get automatic email notification messages whenever any issues under the umbrella are added, resolved, or modified.

## References

[1] 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
Via Marco dell'Arpa 8/B
I-43121 Parma, Italy
Email: info@bugseng.com
Web: http://bugseng.com
Tel.: +39 0521 461640

bugseng

no shortcuts,
no compromises,
no excuses:
software verification done right