Developing high-quality software is tough. ECLAIR is designed to help development, QA, and safety teams reach their quality goals

Coverage of ECSS-Q-ST-80C

1 Introduction to ECSS-Q-ST-80C Rev. 2:2025

ECSS-Q-ST-80C Rev. 2:2025, "Space product assurance — Software product assurance," is a standard issued by the *European Cooperation for Space Standardization* (ECSS) [4]. It defines software product assurance requirements for the development, verification, validation, maintenance, configuration management, and assurance of software used in space systems.

ECSS-Q-ST-80C belongs to the ECSS family of standards and complements ECSS-E-ST-40C [3], the ECSS software engineering standard, by covering the product assurance aspects integrated into the software engineering processes. The prescriptions of ECSS-Q-ST-80C concern, among other things, software product assurance planning and reporting, software dependability and safety, handling of critical software, quality models, process metrics, coding standards, verification and validation, automatic code generation, maintenance, and software security assurance.

The first edition of ECSS-Q-ST-80C was published in 2009. It was revised in 2017 and subsequently replaced by ECSS-Q-ST-80C Rev. 2, published in 2025. The current revision refines and extends previous editions and includes an enhanced treatment of software security aspects.

ECSS-Q-ST-80C tailors the rigor of software product assurance activities according to software criticality and, in Rev. 2, also according to software security assurance and sensitivity. This makes the standard suitable for projects having very different levels of mission criticality, from support software to software whose malfunction can affect mission success, system integrity, dependability, security, or safety.

1.1 Role of ECLAIR in Ensuring Compliance with ECSS-Q-ST-80C

The ECLAIR Software Verification Platform can be used to facilitate compliance with several objectives of ECSS-Q-ST-80C Rev. 2:2025, especially where the standard requires or benefits from:

- well-defined software development and assurance procedures;
- requirements traceability;
- architectural constraints and interface enforcement;

- coding standards and language subsets;
- static analysis, control-flow analysis and data-flow analysis;
- software metrics and quality measurement;
- analysis of changes and regression impact;
- evidence for verification and validation activities; and
- controlled and justified use of software tools in assurance activities.

In particular, ECSS-Q-ST-80C contains explicit requirements and expected outputs related to software product assurance plans, compliance matrices, quality models, verification and validation documentation, software problem reporting, dependability and safety analysis, handling of critical and security sensitive software, coding standards, and software security analysis. The integration of ECLAIR in the development and assurance workflow can provide substantial support in producing, justifying and maintaining the evidence required by those activities.

2 ECLAIR Coverage of ECSS-Q-ST-80C Clauses

The following table summarizes selected clauses of ECSS-Q-ST-80C Rev. 2:2025 for which ECLAIR, suitably instantiated with the appropriate package, can be used to ensure compliance or to facilitate the achievement of compliance.

The table is intentionally conservative: only selected clauses are listed. For the listed clauses, an entry in the ECLAIR column means that ECLAIR can provide direct technical support or useful objective evidence; clauses not listed, or listed without an entry, mainly depend on organizational, managerial, contractual, review, testing, or process activities outside the scope of ECLAIR.

Summary of ECLAIR coverage of ECSS-Q-ST-80C Rev. 2:2025 clauses

Clause	Requirement	ECLAIR
5.2.1.5	Compliance matrix documenting conformance with applicable software product assurance requirements, with references to the documents containing the expected outputs	a
5.2.2.2	Product assurance reporting based on measured properties, verifications undertaken, problems detected, and problems resolved	b
5.2.7.1	Use of quality models to specify software quality requirements	c
5.2.7.2	Quality model characteristics including reliability, maintainability, safety suitability, security, efficiency, portability, and software development effectiveness	c
6.2.2.2	Software dependability and safety analysis of software products to determine the criticality of software components	d, e
6.2.2.5	Reporting on implementation and verification of software dependability and safety analysis recommendations	d, e
6.2.2.6	Update of software dependability and safety analysis at each software development milestone	d, e
6.2.2.7	Provision of software-level analysis results for integration into system-level dependability and safety analyses	d, e
6.2.2.10	If failure propagation or shared resources can cause failures of higher criticality components, all involved components are classified at the highest criticality category	e
6.2.3.2	Definition and application of measures for handling critical software	e, f

continued

Summary of ECLAIR coverage of ECSS-Q-ST-80C Rev. 2:2025 (continued)

Clause	Requirement	ECLAIR
6.2.3.4	Regression testing of critical software after changes in platform hardware functionality or tools affecting executable code generation	g
6.2.5.1	Use of metrics to manage development and assess the quality of development processes	h
6.2.5.2	Regular collection, storage, and analysis of process metrics	h
6.2.6.5	Specific verification of software containing deactivated code	i
6.2.6.6	Specific verification of software containing configurable code	g, i
6.2.8.5	Verification of adherence to modelling standards	j
6.2.8.6	Application of coding requirements to automatically generated code, unless manual modification is unnecessary	j, k
6.2.9.1	Inclusion of software security assurance in the software product assurance plan	l
6.2.9.2	Performance of a software security analysis to determine the sensitivity of software components	l, m
6.2.9.3	Identification of methods and techniques for software security analysis	l, m
6.2.9.4	Engineering measures to reduce the number of security sensitive components and mitigate their risks	e, l, m
6.2.9.5	Reporting on implementation and verification of software security analysis recommendations	l, m
6.2.9.6	Update of software security analysis at each software development milestone	l, m
6.2.9.7	Provision of software security analysis results for integration into system-level security analyses	l, m
6.2.10.1	Measures to avoid propagation of failures, including those caused by deliberate action, between software components	e, l
6.2.10.2	Additional justified measures for security sensitive software, e.g. secure coding practices and static security testing	l, m, n
6.2.10.3	Regression testing of security sensitive software after relevant hardware, tool, or environment changes	g
6.3.2.4	Technical specification to include non-functional requirements including safety, security, configuration management, metrication, and verification and validation	a, c
6.3.2.5	Agreement on methods and tools for agreeing requirements and approving changes	a
6.3.3.2	Definition and application of mandatory and advisory design standards	e, j
6.3.3.4	Verification of adherence to design standards	e, j
6.3.4.1	Coding standards, including security, naming conventions, and commentary rules, to be specified and observed	n
6.3.4.2	Coding standards to be consistent with product quality requirements	c, n
6.3.4.3	Identification, in the product assurance plan, of the tools used to implement and check conformance with coding standards	n, o
6.3.4.4	Review of coding standards with the customer to ensure consistency with product quality and security requirements	c, n, o
6.3.4.6	Definition of measurements, criteria, and tools to ensure that software code meets quality and security requirements	h, n, o
6.3.5.7	Verification of test coverage of configurable code in each tested configuration	g
6.3.9.4	Maintenance plans and procedures to include quality and security measures	g, h, l, n

a ECLAIR service `B.REQMAN` allows ensuring that all code is forward and backward traceable to

documented requirements, including dependability requirements, safety requirements, security requirements, interface requirements, and verification requirements. `B.REQMAN` also allows tracing code to tests and back. This is particularly useful when producing and maintaining compliance matrices and controlled requirement baselines.

b ECLAIR can automatically compute and report numerous measurable software properties, including metrics and static-analysis results, and can therefore provide objective evidence for reports based on measured properties, verifications undertaken, and detected issues.

c ECLAIR supports the practical application of software quality models by checking coding-rule compliance, computing software metrics, and producing evidence related to qualities such as reliability, maintainability, security, efficiency, and portability.

d ECLAIR provides static verification capabilities that can support software dependability and safety analyses by identifying defects, anomalous constructs, data-flow issues, control-flow issues, and violations of language subsets and coding standards.

e ECLAIR service `B.INDEPENDENCE` allows the formal specification and systematic checking of software architectural constraints, including layering constraints, interface usage constraints, restrictions on access to shared resources, and other interaction constraints among software components. This is especially useful for analysing failure propagation, segregation, coexistence of software with different criticalities or sensitivities, and controlled use of shared resources.

f Compliance with coding standards and architectural constraints, together with static verification and metrics, can be used to help define and justify project-specific measures for the handling of critical software.

g ECLAIR analyzes the exact build used to produce the software system. It can therefore support change-impact analysis and regression-related assessment when hardware, tools, configuration options, or relevant parts of the target environment change. Differential reporting with respect to previous analyses further helps assess the impact of such changes.

h ECLAIR automatically computes numerous software metrics and can contribute to the collection and analysis of quantitative information used to manage development processes and assess software quality.

i ECLAIR can help in the analysis of deactivated code, unreachable code, conditional compilation, and configurable code, and can therefore contribute to the specific verification activities required for such code. As usual, this support is partial and complements, rather than replaces, testing and review activities.

j For model-based development workflows that generate source code, ECLAIR can be applied to the generated code and can therefore help verify adherence to the coding and design rules that are applicable to that generated code. It does not, by itself, verify the semantics of the source model.

k Clause 6.3.4 applies to automatically generated code. Accordingly, whenever ECLAIR is used to verify the generated source code against applicable coding standards and selected static-analysis checks, it can contribute to compliance with clause 6.2.8.6.

l ECLAIR contributes to software security assurance through a combination of requirements traceability, architectural-constraint checking, coding-standard enforcement, static verification, and analysis of the exact build used to generate the software system.

m ECLAIR can support software security analysis, in particular where that analysis includes requirements analysis, design analysis, and code analysis. It does not replace broader security engineering activities, system-level security analyses, fuzzing, dynamic testing, penetration testing, or vulnerability management processes.

- n ECLAIR can be used to verify compliance with secure coding practices and language subsets, such as the MISRA coding standards, and can therefore help enforce and verify those coding standards that are introduced to satisfy software quality and software security requirements.
- o The *ECLAIR Qualification Kits* can provide substantial support when the project needs documented justification for the use of ECLAIR as one of the tools employed to implement and check compliance with coding standards and related project rules.

2.1 MISRA C:2025

MISRA C:2025 [7] is the software development C subset developed by MISRA that is a de facto standard for safety-, life-, security-, and mission-critical embedded applications in many industries, including aerospace, railway, medical, telecommunications and others. MISRA C:2025, which allows coding MISRA-compliant applications in subsets of C90, C99, C11 and C18, is supported, along with all previous versions of MISRA C, by the ECLAIR package called "MC".

2.2 MISRA C++:2023

MISRA C++:2023 [6] is the software development C++ subset developed by MISRA, which is a de facto standard for safety-, life-, and mission-critical embedded applications in many industries including aerospace, railway, medical, telecommunications and others. MISRA C++:2023 completely supersedes MISRA C++:2008 [8], the previous edition of the coding standard, which is still used by many legacy projects. MISRA C++:2023 and MISRA C++:2008 are supported by the ECLAIR package called "MP".

2.3 BARR-C:2018

The *Barr Group's Embedded C Coding Standard*, BARR-C:2018 [2], is, for coding standards used by the embedded system industry, second only in popularity to MISRA C. BARR-C:2018 guidelines include 64 guidelines dealing with language subsetting and project management as well as 79 guidelines concerning programming style. For projects in which a MISRA compliance requirement is not (yet) present, the adoption of BARR-C:2018 is a major improvement with respect to the situation where no coding standards and no static analysis is used. The adoption of the stylistic subset of BARR-C:2018 (79 out of 143 rules) can be part of complying with the MISRA requirement that a consistent programming style is adopted and systematically used as part of the software development process. Moreover, complying with BARR-C:2018, besides avoiding many dangerous bugs, entails compliance with a non-negligible subset of MISRA C:2012 [1]. ECLAIR support for BARR-C:2018 has no equals on the market: it is included in all ECLAIR packages, including the affordable package "B".

2.4 HIS and Other Source Code Metrics

Source code metrics are recognized by many software process standards (and from MISRA) as providing an objective foundation to efficient project and quality management. One well known set of metrics has been defined by HIS (Herstellerinitiative Software, an interest group set up by Audi, BMW, Daimler, Porsche and Volkswagen).

The *HIS source code metrics* [5], while well established, include some metrics that are obsolete and miss others that are required or recommended by software process standards, such as those that allow estimating function coupling. For this reason, ECLAIR supplements HIS source code metrics with numerous other metrics that allow software quality to be assessed in terms of complexity, testability, readability, maintainability and so forth. Keeping track of these metrics also provides an effective and objective method to assess the quality of the software development process. The full set of metrics is available in all ECLAIR packages.

2.5 ECLAIR Support for Requirements Traceability

ECSS-Q-ST-80C places considerable emphasis on documented outputs, controlled baselines, and demonstrable compliance of the software product assurance process.

ECLAIR service `B.REQMAN` allows ensuring that all code is forward and backward traceable to documented requirements, including functional requirements, interface requirements, dependability requirements, safety requirements, and security requirements. `B.REQMAN` also allows tracing code to tests and back. The integrated requirements management tool makes ECLAIR a cost-effective, complete solution for requirements-based development and verification.

2.6 ECLAIR Support for Architecture and Interface Control

ECSS-Q-ST-80C requires disciplined development processes, controlled interfaces, and appropriate treatment of critical and security sensitive software. For software systems, this typically entails the precise specification of software architecture, interfaces, dependencies, and allowed interactions among software items.

ECLAIR service `B.INDEPENDENCE` allows the formal specification and systematic checking of software architectural constraints, e.g., to enforce layering constraints, prevent bypassing of software interfaces, regulate access to shared resources, and support segregation arguments among software components of different criticality or sensitivity. This is particularly useful in projects where independence, containment of failure propagation, or controlled interaction among software items must be justified.

2.7 ECLAIR Support for Verification, Validation and Quality Measurement

ECSS-Q-ST-80C explicitly addresses software verification and validation, and requires the use of measured properties within software product assurance reporting and planning.

ECLAIR provides static verification capabilities based on coding-standard checking, control-flow analysis, data-flow analysis, abstract interpretation, and a broad set of software metrics. These capabilities support the verification of software properties before testing is reached, thereby helping to prevent the introduction of defects and to detect classes of defects early in the lifecycle.

ECLAIR also provides numerous metrics related to complexity, size, structure, documentation quality, and other measurable software properties. These metrics can contribute to the metrication activities required by ECSS-Q-ST-80C and can be used as objective evidence in software product assurance reports and milestone reviews.

2.8 ECLAIR Support for Software Security Assurance

ECSS-Q-ST-80C Rev. 2 strengthens the treatment of software security, including software security analysis and the handling of security sensitive software.

ECLAIR contributes to software security assurance in several ways.

First, compliance checking against MISRA coding standards and other coding rules helps prevent many classes of defects that are relevant to both safety and security.

Second, architectural-constraint checking through `B.INDEPENDENCE` helps enforce isolation, layering, and controlled use of interfaces, thus supporting the containment of vulnerabilities and the reduction of unintended interactions.

Third, requirements traceability through `B.REQMAN` helps ensure that security requirements are properly allocated, implemented, and linked to the corresponding verification activities.

Finally, the ability of ECLAIR to analyze the exact build used for the software system, and to compare analyses across versions, helps support change-impact assessment and the maintenance of security-related evidence throughout the lifecycle.

3 ECLAIR Qualification in Compliance with ECSS-Q-ST-80C

ECSS-Q-ST-80C requires that the tools used to implement and check conformance with coding standards be identified in the software product assurance plan, while ECSS-E-ST-40C requires the selection of software tools to be justified and supported by evidence of suitability, and explicitly includes static-analysis tools among the relevant software tools. Furthermore, ECSS guidance material discusses tool qualification and confidence in the qualification status of tools used for development, verification, or validation, especially when project activities are reduced, automated, or replaced by tool output that is not otherwise verified.

Accordingly, the fact that ECLAIR has been independently assessed and certified by TÜV SÜD with respect to several functional-safety standards up to the highest criticality levels is relevant for ECSS-based projects, since it can contribute useful supporting evidence that ECLAIR is a mature and independently assessed tool for high-assurance development and verification contexts.

The ECLAIR Qualification Kits for ECSS-Q-ST-80C provide further help to safety teams in charge of qualifying ECLAIR for use in safety-related projects where the dependence on the tool operational environment has to be taken into account: the kits contain documents, test suites, procedures and automation facilities that can be used by the customer to independently obtain all the required confidence-building evidence. BUGSENG also offers the ECLAIR Qualification Service, whereby qualified BUGSENG personnel undertakes almost all the qualification effort.

4 The Bigger Picture

ECLAIR is very flexible and highly configurable: it supports all kinds of software development workflows and environments.

ECLAIR is fit for use in mission- and safety-critical software projects: it has been designed from the outset to exclude configuration errors that would undermine the significance of the obtained results.

ECLAIR is developed in a rigorous way and carefully checked with extensive internal test suites (tens of thousands of test cases) and industry-standard validation suites.

ECLAIR is based on solid scientific research results and on the best practices of software development.

ECLAIR's unique features and BUGSENG's strong commitment to the customer, allow for a smooth transition to ECLAIR from any other tool.

BUGSENG's quality system has been certified by TÜV Italia (TÜV SÜD Group) to comply with the requirements of UNI EN ISO 9001:2015 for the "Design, development, maintenance and support of tools for software verification and validation" (IAF 33).

BUGSENG is an Arm's Functional Safety Partner, and is thus recognized as a partner who can reliably support their customers with industry leading functional safety products and services.

For More Information

BUGSENG srl
Via Fiorentina 214/C
I-56121 Pisa, Italy
Email: info@bugseng.com
Web: http://bugseng.com

bugseng

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

References

[1] R. Bagnara, M. Barr, and P. M. Hill. BARR-C:2018 and MISRA C:2012 (with Amendment 2): Synergy between the two most widely used C coding standards. In DESIGN&ELEKTRONIK, editor, *embedded world Conference 2021 DIGITAL — Proceedings*, pages 378–391, Nuremberg, Germany, 2021. WEKA FACHMEDIEN, Richard-Reitzner-Allee 2, 85540 Haar, Germany.

[2] M. Barr. *BARR-C:2018 — Embedded C Coding Standard*. Barr Group, www.barrgroup.com, 2018.

[3] ECSS Secretariat. *ECSS-E-ST-40C Rev. 1 — Software*. European Cooperation for Space Standardization (ECSS), Noordwijk, The Netherlands, April 2025. Active Standard.

[4] ECSS Secretariat. *ECSS-Q-ST-80C Rev. 2 — Software product assurance*. European Cooperation for Space Standardization (ECSS), Noordwijk, The Netherlands, April 2025. Active Standard.

[5] H. Kuder et al. HIS source code metrics. Technical Report HIS-SC-Metriken.1.3.1-e, Herstellerinitiative Software, April 2008. Version 1.3.1.

[6] MISRA. *MISRA C++:2023 — Guidelines for the use of C++17 in critical systems*. The MISRA Consortium Limited, Norwich, Norfolk, NR3 1RU, UK, October 2023.

[7] MISRA. *MISRA C:2025 — Guidelines for the use of the C language in critical systems*. The MISRA Consortium Limited, Norwich, Norfolk, NR3 1RU, UK, March 2025.

[8] Motor Industry Software Reliability Association. *MISRA C++:2008 — Guidelines for the use of the C++ language in critical systems*. MIRA Limited, Nuneaton, Warwickshire CV10 0TU, UK, June 2008.