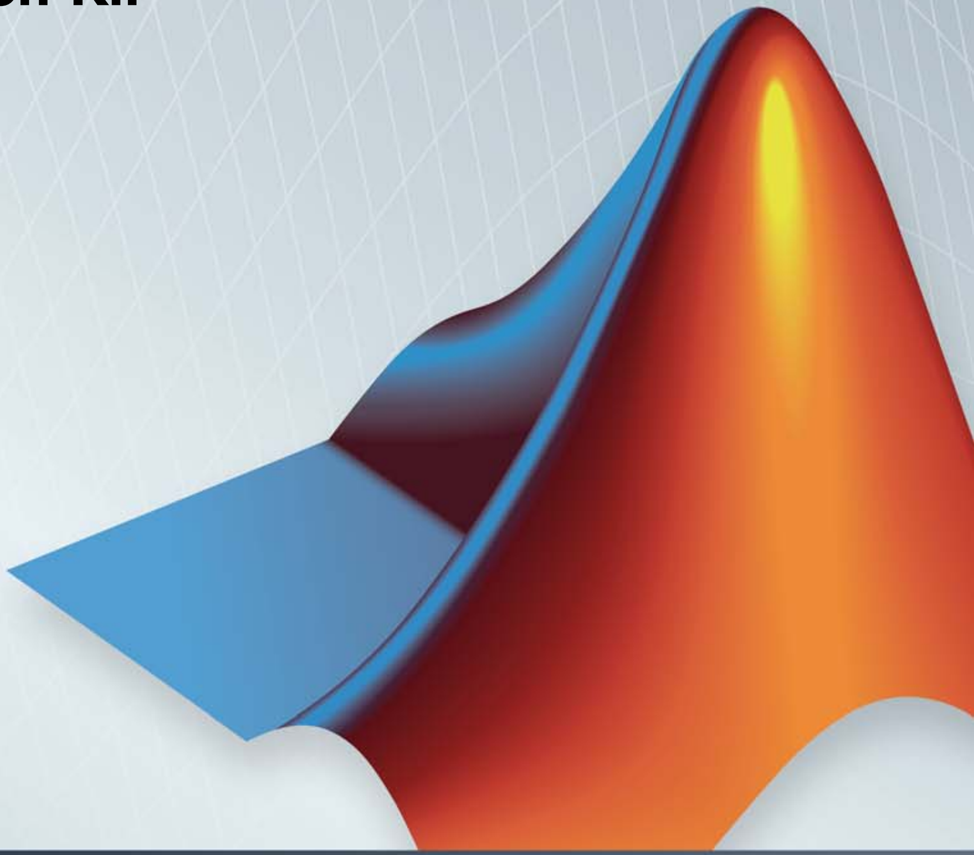
IEC Certification Kit

User's Guide

R2013a



MATLAB® & SIMULINK®

MathWorks®

Revision History

March 2009	Online only	New for Version 1.0 (Applies to Releases 2007a+, 2008a, 2008b, 2009a)
September 2009	Online only	Revised for Version 1.1 (Applies to Releases 2008a, 2008b, 2009a, 2009a+, 2009b)
March 2010	Online only	Revised for Version 1.2 (Applies to Release 2010a)
September 2010	Online only	Revised for Version 1.3 (Applies to Releases 2009bSP1, R2010a, 2010b)
April 2011	Online only	Revised for Version 1.4 (Applies to Releases 2010bSP1, 2011a)
September 2011	Online only	Revised for Version 2.0 (Applies to Release 2011b)
March 2012	Online only	Revised for Version 2.1 (Applies to Release 2012a)
September 2012	Online only	Revised for Version 3.0 (Applies to Releases 2010bSP2, 2012b)
March 2013	Online only	Revised for Version 3.1 (Applies to Release 2013a)

Contents

1 Getting Started with IEC Certification Kit

Product Description 1-2
Key Features 1-2

Required Knowledge 1-3

IEC Certification Kit Component Summary 1-4

Access Certification Artifacts for Embedded Coder . . . 1-6

Access Certification Artifacts for Simulink PLC Coder 1-8

Access Certification Artifacts for Simulink Design Verifier 1-9

Access Certification Artifacts for Simulink Verification and Validation 1-10

Access Certification Artifacts for Polyspace Client for C/C++ and Polyspace Server for C/C++ 1-11

Access Supporting Artifacts for ISO 26262 1-13

Access Supporting Artifacts for EN 50128 1-14

Access Supporting Artifacts for IEC 61508 1-15

Limitations 1-16

ISO 26262 1-17

What Is ISO 26262? 1-17
ISO 26262 Compliance Considerations 1-18
ISO 26262 Tool Qualification Considerations 1-18

IEC 61508 .... **1-19**
What Is IEC 61508? 1-19
IEC 61508 Compliance Considerations 1-20
IEC 61508 Tool Certification Considerations 1-20

EN 50128 .... **1-22**
What Is EN 50128? 1-22
EN 50128 Software Tool Considerations 1-22

IEC 61511 .... **1-23**
What Is IEC 61511? 1-23

2 Reference Workflows

Embedded Coder Reference Workflow Overview **2-2**

Simulink PLC Coder Reference Workflow Overview .. **2-5**

Simulink Design Verifier Reference Workflow Overview **2-8**

Simulink Verification and Validation Reference Workflow Overview **2-10**

Polyspace Client/Server for C/C++ Reference Workflow Overview **2-12**

3 Certification Process

Define Certification Objectives and Requirements 3-2

Certify or Qualify Software Tools 3-3

Document Evidence of Using Tools Within Referenced Workflows 3-4
ISO 26262 Tool Qualification Artifacts 3-4
IEC 61508 Tool Certification Artifacts 3-5
EN 50128 Tool Certification Artifacts 3-8

4 Validate Software Tools

Software Tool Validation 4-2

Run Test Cases and Procedures for Embedded Coder 4-3

Run Test Cases and Procedures for Simulink Verification and Validation 4-4

Run Test Cases and Procedures for Polyspace Client/Server for C/C++ 4-6

5 Access and Manage Certification Artifacts

Access Artifacts Using the Certification Artifacts Explorer 5-2
Certification Artifacts in the IEC Certification Kit Product 5-2

What Is a Certification Package? 5-2
How To Access Certification Artifacts 5-3

Manage Artifacts Using the Certification Artifacts Explorer **5-5**

Delete Certification Packages from the Certification Artifacts Explorer **5-7**

Certification Artifacts Explorer Limitations **5-8**

6 Support Certification-Related Development Activities

Generate a Traceability Matrix **6-2**
About Traceability Matrices 6-2
Prerequisites for Generating a Traceability Matrix 6-3
How to Generate a Traceability Matrix 6-4

Add Comments to a Traceability Matrix **6-6**
Requirements for Adding Comments to a Traceability Matrix 6-6
How To Retain Comments 6-7

Traceability Matrix Limitations **6-8**

Display Bug Reports **6-9**

7 Functions — Alphabetical List

8 Model Advisor Checks

IEC Certification Kit Checks **8-2**
Display bug reports using IEC Certification Kit checks ... **8-3**
Display bug reports for Simulink Verification and Validation **8-4**
Display bug reports for Simulink Design Verifier **8-5**
Display bug reports for Simulink PLC Coder **8-6**
Display bug reports for IEC Certification Kit **8-7**
Display bug reports for Polyspace products for C/C++ **8-8**
Display bug reports for Embedded Coder **8-8**

Getting Started with IEC Certification Kit

- "Product Description" on page 1-2
- "Required Knowledge" on page 1-3
- "IEC Certification Kit Component Summary" on page 1-4
- "Access Certification Artifacts for Embedded Coder" on page 1-6
- "Access Certification Artifacts for Simulink® PLC Coder™" on page 1-8
- "Access Certification Artifacts for Simulink® Design Verifier™" on page 1-9
- "Access Certification Artifacts for Simulink® Verification and Validation™" on page 1-10
- "Access Certification Artifacts for Polyspace® Client™ for C/C++ and Polyspace® Server™ for C/C++ " on page 1-11
- "Access Supporting Artifacts for ISO 26262" on page 1-13
- "Access Supporting Artifacts for EN 50128" on page 1-14
- "Access Supporting Artifacts for IEC 61508" on page 1-15
- "Limitations" on page 1-16
- "ISO 26262" on page 1-17
- "IEC 61508" on page 1-19
- "EN 50128" on page 1-22
- "IEC 61511" on page 1-23

Product Description

Qualify code generation and verification tools for ISO 26262 and IEC 61508 certification

IEC Certification Kit provides tool qualification artifacts, certificates, and test suites, and generates traceability matrices. The kit helps you qualify code generation and verification products and streamline certification of your embedded systems to ISO® 26262, IEC 61508, EN 50128, and related functional-safety standards. Certificates and assessment reports from the certification authority TÜV SÜD are included in the kit for the supported products and standards.

IEC Certification Kit provides ISO 26262 tool classification and qualification work products, together with test suites. It includes templates that let you adapt the work products to meet specific project needs. You can generate project-specific artifacts, including traceability matrices covering requirements, models, and generated code. Project- and product-specific artifacts can be combined to produce a complete ISO 26262 tool qualification package for embedded system certification.

Key Features

- TÜV SÜD certificates and reports for supported Simulink® products
- TÜV SÜD certificates and reports for supported Polyspace® products
- ISO 26262 tool classification and qualification work products and test suites
- Traceability matrix generation covering requirements, models, and generated code
- Customizable templates for delivering documentation to certification authorities
- Artifacts explorer for navigating and viewing artifacts for each supported product and standard
- Checks for tool-associated bug reports

Required Knowledge

Before using the IEC Certification Kit product, make sure that you have:

- Knowledge about developing safety-related software.
- Knowledge of the applicable safety standard:
  - ISO 26262 *Road vehicles - Functional safety*
  - IEC 61508 *Functional safety of electrical/electronic/programmable electronic safety-related systems*
  - EN 50128 *Railway Applications - Communications, Signalling and Processing Systems - Software for Railway Control and Protection Systems*
  - IEC 61511 *Functional safety - Safety Instrumented Systems for the process industry sector*
- Experience with MathWorks® products that you use to develop, verify, or validate software for systems that are required to comply with the applicable standard.

If you have an Embedded Coder® license, also review the following information:

- “ISO 26262 Standard” in the Embedded Coder documentation
- “IEC 61508 Standard” in the Embedded Coder documentation

IEC Certification Kit Component Summary

The IEC Certification Kit product includes the following certification artifacts and tools:

- Certification and qualification evidence
- Documents and templates
- Tools for certification-related development activities
- Tools for managing certification artifacts
- Test cases and test procedures to support tool validation

The certification artifacts and tools support you when using the following MathWorks products in the context of the ISO 26262, IEC 61508, EN 50128, or IEC 61511 standards:

- Embedded Coder
- Simulink PLC Coder™
- Simulink Design Verifier™
- Simulink Verification and Validation™
- Polyspace Client™ for C/C++; Polyspace Server™ for C/C++

Specific versions of the preceding MathWorks products have been certified or prequalified by TÜV SÜD, a German-based certification authority, according to one or more of the above mentioned standards.

For a list of products supported by the IEC Certification Kit for each MathWorks release, see Technical Solution 1-JJ3OM1.

The IEC Certification Kit product contains certification artifacts to document compliance with the respective standards. The applicant can submit certification artifacts, or derivatives thereof, as evidence of compliance with ISO 26262-6, ISO 26262-8, IEC 61508-3, EN 50128, or IEC 61511-1. Note that artifacts included in the kit are not accessible from the MathWorks web site.

The IEC Certification Kit product provides the following capabilities to support certification-related development activities:

- Generation of traceability matrices covering model requirements, model objects, and generated code (see "Generate a Traceability Matrix" on page 6-2).
- Model Advisor checks for tool-associated bug reports (see "Display Bug Reports" on page 6-9).

The IEC Certification Kit product provides an Artifacts Explorer, a tool for accessing and managing work products created during the development of high-integrity systems, including certification artifacts (see "Artifact Management").

The IEC Certification Kit product provides test procedures that can be used to automate tool validation tests for Embedded Coder, Simulink Verification and Validation, and Polyspace Client/Server for C/C++ (see "Tool Validation Automation").

Note The `rights.txt` file, located at *`matlabroot`*`/toolbox/qualkits/iec`, describes allowed uses of the IEC Certification Kit product.

Access Certification Artifacts for Embedded Coder

TÜV SÜD has certified specific versions of the Embedded Coder and product for use in development processes that are required to comply with ISO 26262, IEC 61508, or EN 50128. These product versions are also prequalified according to ISO 26262-8 for Automotive Safety Integrity Levels ASIL A through ASIL D.

The IEC Certification Kit product contains certification artifacts for the following versions of the Embedded Coder product:

Version 6.4 (R2013a)

Previous releases of the Embedded Coder product are certified or prequalified. For supporting certification artifacts, see previous releases of the IEC Certification Kit product.

Note The Embedded Coder product was not developed using an IEC 61508 certified process.

Open the Artifacts Explorer to access the certification artifacts. Alternatively, on the MATLAB® command line, type `certkitiec`. The certification artifacts are located in **Embedded Coder > r2013a**.

Details on the certification artifacts are in the certificate reports.

Component	File
Certificate	`certkitiec_ecoder_certificate.pdf`
Certificate Report	`certkitiec_ecoder_certreport.pdf`
"Embedded Coder Reference Workflow Overview" on page 2-2	`certkitiec_ecoder_workflow.pdf`
Conformance Demonstration Template	`certkitiec_ecoder_cdt.docx/.pdf`

Component	File
ISO 26262 Tool Qualification Package	`certkitiec_ecoder_tqp.docx/.pdf`

Test Procedure / Test Cases	`certkitiec_ecoder_tests.m`
	`certkitiec_ecoder_modelList.m`
	`/tests/*`
	`/outputs/*`
	`/baseline/*`

Access Certification Artifacts for Simulink PLC Coder

TÜV SÜD certified specific versions of the Simulink PLC Coder product for use in development processes that are required to comply with IEC 61508 or IEC 61511.

The IEC Certification Kit product contains certification artifacts for the following versions of the Simulink PLC Coder product:

Version 1.5 (R2013a)

Previous releases of the Simulink PLC Coder product are certified. For supporting certification artifacts, see previous releases of the IEC Certification Kit product.

Note The Simulink PLC Coder product was not developed using an IEC 61508 certified process.

Open the Artifacts Explorer to access the certification artifacts. Alternatively, on the MATLAB command line, type `certkitiec`. The certification artifacts are located in **Simulink PLC Coder > r2013a**.

Details on the certification artifacts are in the certificate reports.

Component	File
Certificate	`certkitiec_plccoder_certificate.pdf`
Certificate Report	`certkitiec_plccoder_certreport.pdf`
"Simulink® PLC Coder™ Reference Workflow Overview" on page 2-5	`certkitiec_plccoder_workflow.pdf`
Conformance Demonstration Template	`certkitiec_plccoder_cdt.docx/.pdf`

Access Certification Artifacts for Simulink Design Verifier

TÜV SÜD has certified specific versions of the Simulink Design Verifier product for use in development processes that are required to comply with ISO 26262, IEC 61508, or EN 50128. These product versions are also prequalified according to ISO 26262-8 for Automotive Safety Integrity Levels ASIL A through ASIL D.

The IEC Certification Kit product contains certification artifacts for the following versions of the Simulink Design Verifier product:

Version 2.4 (R2013a)

Note The Simulink Design Verifier product was not developed using an IEC 61508 certified process.

Open the Artifacts Explorer to access the certification artifacts. Alternatively, on the MATLAB command line, type `certkitiec`. The certification artifacts are located in **Simulink Design Verifier > r2013a**.

Details on the certification artifacts are in the certificate reports.

Component	File
Certificate	`certkitiec_sldv_certificate.pdf`
Certificate Report	`certkitiec_sldv_certreport.pdf`
"Simulink® Design Verifier™ Reference Workflow Overview" on page 2-8	`certkitiec_sldv_workflow.pdf`
Conformance Demonstration Template	`certkitiec_sldv_cdt.docx/.pdf`
ISO 26262 Tool Qualification Package	`certkitiec_sldv_tqp.docx/.pdf`

Access Certification Artifacts for Simulink Verification and Validation

TÜV SÜD has certified specific versions of the Simulink Verification and Validation product for use in development processes that are required to comply with ISO 26262, IEC 61508, or EN 50128. These product versions are also prequalified according to ISO 26262-8 for Automotive Safety Integrity Levels ASIL A through ASIL D.

The IEC Certification Kit product contains certification artifacts for the following versions of the Simulink Verification and Validation product:

Version 3.5 (R2013a)

Note The Simulink Verification and Validation product was not developed using an IEC 61508 certified process.

Open the Artifacts Explorer to access the certification artifacts. Alternatively, on the MATLAB command line, type `certkitiec`. The certification artifacts are located in **Simulink Verification and Validation > r2013a**.

Details on the certification artifacts are in the certificate reports.

Component	File
Certificate	`certkitiec_slvnv_certificate.pdf`
Certificate Report	`certkitiec_slvnv_certreport.pdf`
"Simulink® Verification and Validation™ Reference Workflow Overview" on page 2-10	`certkitiec_slvnv_workflow.pdf`
Conformance Demonstration Template	`certkitiec_slvnv_cdt.docx/.pdf`
ISO 26262 Tool Qualification Package	`certkitiec_slvnv_tqp.docx/.pdf`

Test Procedure / Test Cases	`certkitiec_slvnv_tests*.rpt/.xls`
	`/tests/*`
	`/outputs/*`

Access Certification Artifacts for Polyspace Client for C/C++ and Polyspace Server for C/C++

TÜV SÜD certified specific versions of the Polyspace Client for C/C++ and the Polyspace Server for C/C++ products for use in development processes that are required to comply with ISO 26262, IEC 61508, or EN 50128. These product versions are also prequalified according to ISO 26262-8 for Automotive Safety Integrity Levels ASIL A through ASIL D.

The IEC Certification Kit product contains certification artifacts for the following versions of the Polyspace Client for C/C++ and the Polyspace Server for C/C++ products:

Version 8.5 (R2013a)

Previous releases of the Polyspace products are certified or prequalified. For supporting certification artifacts, see previous releases of the IEC Certification Kit product.

Note The Polyspace Client for C/C++ and the Polyspace Server for C/C++ products were not developed using an IEC 61508 certified process.

Open the Artifacts Explorer to access the certification artifacts. Alternatively, on the MATLAB command line, type `certkitiec`. The certification artifacts are located in **Polyspace Client/Server for C/C++ > r2013a**.

Component	File
Certificate	`certkitiec_polyspace_certificate.pdf`
Certificate Report	`certkitiec_polyspace_certreport.pdf`
"Polyspace Client/Server for C/C++ Reference Workflow Overview" on page 2-12	`certkitiec_polyspace_workflow.pdf` `certkitiec_polyspace_sqo.pdf`
Conformance Demonstration Template	`certkitiec_polyspace_cdt.docx/.pdf`

Component	File
ISO 26262 Tool Qualification Package	`certkitiec_polyspace_tqp.docx/.pdf`
Test Procedure / Test Cases	`/tests/*` (including `/tests/certkitiec_polyspace_tests.bat/.sh`) `/outputs/*`

Access Supporting Artifacts for ISO 26262

The IEC Certification Kit product contains the following artifacts to support ISO 26262 compliance:

- *Model-Based Design for ISO 26262* — Provides suggestions for leveraging MathWorks tools and workflows for Model-Based Design when applying the ISO 26262–6 and ISO 26262–8 standards.
- *Software Tool Inventory* — Provides a template for listing the software tools used in the project under consideration and their corresponding tool classification/qualification documentation.

Open the Artifacts Explorer to access the certification artifacts. Alternatively, on the MATLAB command line, type `certkitiec`. The supporting artifacts are located in **Supporting Artifacts**.

Component	File
Model-Based Design for ISO 26262	`certkitiec_mbd_iso26262.docx/.pdf`
Software Tool Inventory	`certkitiec_tools.docx/.pdf`

Access Supporting Artifacts for EN 50128

The IEC Certification Kit product contains the following artifact to support EN 50128 compliance:

- *Model-Based Design for EN 50128* — Provides suggestions for leveraging MathWorks tools and workflows for Model-Based Design when applying the EN 50128 standard.
- *Software Tool Inventory* — Provides a template for listing the software tools used in the project under consideration and their corresponding tool classification/qualification documentation.

Open the Artifacts Explorer to access the certification artifacts. Alternatively, on the MATLAB command line, type `certkitiec`. The supporting artifacts are located in **Supporting Artifacts**.

Component	File
Model-Based Design for EN 50128	`certkitiec_mbd_en50128.docx/.pdf`
Software Tool Inventory	`certkitiec_tools.docx/.pdf`

Access Supporting Artifacts for IEC 61508

The IEC Certification Kit product contains the following artifact to support IEC 61508 compliance:

- *Model-Based Design for IEC 61508* — Provides suggestions for leveraging MathWorks tools and workflows for Model-Based Design when applying the IEC 61508 standard.
- *Software Tool Inventory* — Provides a template for listing the software tools used in the project under consideration and their corresponding tool classification/qualification documentation.

Open the Artifacts Explorer to access the certification artifacts. Alternatively, on the MATLAB command line, type `certkitiec`. The supporting artifacts are located in **Supporting Artifacts**.

Component	File
Model-Based Design for IEC 61508	`certkitiec_mbd_iec61508.docx/.pdf`
Software Tool Inventory	`certkitiec_tools.docx/.pdf`

Limitations

Neither compliance with nor certification to the applicable safety standard ensure the safety of the software or the system under consideration. However, the applicable safety standard may be considered a state-of-the-art or generally accepted rules of technology (GART) for the development of safety-related systems in your industry. A certification might be used as evidence that state-of-the-art procedures were applied during system development.

ISO 26262

In this section...

"What Is ISO 26262?" on page 1-17

"ISO 26262 Compliance Considerations" on page 1-18

"ISO 26262 Tool Qualification Considerations" on page 1-18

What Is ISO 26262?

ISO 26262 is an international functional safety standard titled *Road vehicles — Functional safety*. ISO 26262 is the adaptation of IEC 61508 to comply with needs specific to the application sector of *E/E systems*[1] within road vehicles.

ISO published the ISO 26262 standard in 2011. It consists of ten parts, referred to as ISO 26262-1 to ISO 26262-10.

Part 2 (ISO 26262-2) *Management of functional safety* specifies the requirements on functional safety management for automotive applications. Part 6 (ISO 26262-6) *Product development: software level* pertains to software development, verification, and validation. It includes guidance for projects using Model-Based Design[2] and code generation. Part 8 (ISO 26262-8) *Supporting processes* addresses multiple cross-functional topics, including the classification and qualification of software tools.

The required degree of rigor for software development, verification, and validation varies, depending on how critical the software is. It is expressed in terms of Automotive Safety Integrity Levels (ASILs) A to D. For example, a measure or technique listed in ISO 26262 might be recommended for ASIL A and ASIL B, and highly recommended for ASIL C and ASIL D.

1. Systems that consists of electrical and electronic elements, including: programmable electronic elements, power supplies, input devices, communication paths, and output devices.

2. Referred to as *model-based development*.

ISO 26262 Compliance Considerations

ISO 26262-2 lays out confirmation measures to be carried out in order to claim compliance with the standard.

ISO 26262 Tool Qualification Considerations

ISO 26262-8 provides a framework for software tool classification and qualification to provide evidence that a software tool is suitable for use when developing safety-related software. In this way, confidence can be achieved in the correct execution of the activities and tasks supported by this tool (see ISO 26262-8, clause 11).

To determine the required level of confidence in a software tool (tool confidence level, TCL), the applicant shall analyze the use cases for the software tool. The analysis determines:

- If a malfunctioning software tool and the erroneous output of the tool can lead to the violation of a safety requirement.
- The probability of preventing or detecting such errors in the output.

The evaluation considers tool-internal measures (for example, monitoring), as well as tool-external measures (for example, guidelines, tests, reviews) that the applicant implements in the development process for the safety-related software.

The required TCL, together with the ASIL of the software developed using the tool, determines whether the tool must be qualified and allows the selection of relevant qualification methods.

Regardless of the tool qualification, the tool user is and remains fully responsible for the safety of the system and its embedded software.

IEC 61508

In this section...

"What Is IEC 61508?" on page 1-19

"IEC 61508 Compliance Considerations" on page 1-20

"IEC 61508 Tool Certification Considerations" on page 1-20

What Is IEC 61508?

IEC 61508 is an international, industry-independent functional safety standard, titled *Functional safety of electrical/electronic/programmable electronic safety-related systems*. The seven parts of the standard (referred to as IEC 61508-1 to IEC 61508-7) were published in 2010.

IEC 61508-3 *Software Requirements* concerns software development, verification, and validation. By constraining the processes used for software development and quality assurance, the intention of the IEC 61508-3 standard is to:

- Reduce the number of errors introduced during software development.
- Increase the number of errors revealed by verification and validation activities.

IEC 61508 is a prescriptive standard, providing detailed lists of techniques and measures with recommendations. The required degree of rigor for software development, verification, and validation varies, depending on how critical the software is. The standard expresses the degree of rigor in terms of Safety Integrity Levels (SILs). For example, IEC-61508-3 might recommend a measure or technique for SIL 1 and 2, and highly recommend it for SIL 3 and 4.

To help with the selection of techniques and measures relevant for a required SIL, annexes A and B of IEC 61508-3 provide software safety integrity tables. The tables list the techniques and measures recommended for each SIL. The standard organizes the tables based on the different software lifecycle phases. IEC 61508-7 *Overview of techniques and measures* provides detailed descriptions of selected measures and techniques.

IEC 61508 Compliance Considerations

IEC 61508 certification confirms that a product or system complies with objectives set by the standard.

You can get IEC 61508 compliance certified by an independent, external certification authority, such as Technischer Überwachungsverein (TÜV) in Germany. Upon granting certification, the certification authority issues a certificate and, if applicable, a certificate report. A certificate report is a technical report that accompanies the certificate. The certificate report documents details of the certification process and constraints for the certificate.

An applicant might self-certify a system. Self-certification requires the applicant to demonstrate IEC 61508 compliance to an internal assessor, without requiring external certification. In this case, aspects of the standard might be relaxed or tightened.

Regardless of how an applicant achieves certification, the applicant shall document compliance with the relevant set of IEC 61508 requirements. For software, the applicant typically creates customized instances of software safety integrity tables. The tables describe how you interpreted and applied each recommended technique and measure for the software under development. If a highly recommended technique or measure is not used, the rationale shall be documented and agreed upon with the certification authority or internal assessor.

The customized software safety integrity tables serve as partial evidence to demonstrate that the objectives of the standard are met. To facilitate certification, the applicant should submit an initial version of the tables early in the software development lifecycle to the certification authority or internal assessor for discussion and approval.

IEC 61508 Tool Certification Considerations

The intention of the IEC 61508 standard is to regulate the development of safety-related systems, not the development of software tools used to design, verify, and validate these systems. However, IEC 61508 includes some requirements on the usage of software tools. In particular, IEC 61508-3, clause 7.4.4 provides requirements for tools used to develop safety-related software, including a tool classification scheme and requirements for tool validation.

IEC 61508-3, table A.3 highly recommends certified tools and translators for safety integrity levels SIL 2 and higher.

Different tool certification approaches have been proposed and pursued in practice. A recent approach is in-context certification of tools. In-context certification is based on a specific workflow or set of workflows to be used when applying the tool to develop or verify software for IEC 61508 compliant or certified applications. For an in-context certification, the certification package includes a reference workflow document in addition to a certificate and certificate report. The applicant shall ensure the tool is used within the workflows referenced and the constraints specified in their respective certificates.

Regardless of the tool certification, the tool user is and remains fully responsible for the safety of the system and its embedded software.

EN 50128

In this section...

"What Is EN 50128?" on page 1-22

"EN 50128 Software Tool Considerations" on page 1-22

What Is EN 50128?

EN 50128 is a European safety standard titled *Railway applications - Communications, signalling and processing systems - Software for railway control and protection systems*. The standard specifies procedures and technical requirements for the development of programmable electronic systems for use in railway control and protection applications. EN 50128, developed by the European Committee for Electrotechnical Standardization (CENELEC), is part of a series of standards that represent the railway application-specific interpretation of the IEC 61508 standard series.

EN 50128 Software Tool Considerations

Requirements for support tools are specified in clause 6.7 of the EN 50128 standard. The objective of this clause is "to provide evidence that potential failures of tools do not adversely affect the integrated toolset output in a safety related manner that is undetected by technical and/or organizational measures outside the tool." (EN 50128:2011).

IEC 61511

What Is IEC 61511?

IEC 61511 is an international functional safety standard titled *Functional safety - Safety Instrumented Systems for the process industry sector*. IEC 61511 has been developed as a process sector implementation of IEC 61508. The standard consists of three parts, referred to as IEC 61511-1 to IEC 61511-3. Part 1 (IEC 61511-1) covers framework, definitions, and system, hardware, and software requirements.

2

Reference Workflows

Embedded Coder Reference Workflow Overview

The *Embedded Coder Reference Workflow* describes a workflow for application-specific verification and validation of models and generated C and C++ code developed using Model-Based Design with production code generation. Users of the Embedded Coder software shall carry out this workflow as part of the overall ISO 26262, IEC 61508, or EN 50128 software safety lifecycle. Model-Based Design enables automatic generation of production-quality code from executable graphical models that you can deploy onto embedded systems. Simulink products from MathWorks have become an accepted standard for Model-Based Design. “Simulink”, “Fixed-Point Designer™”, and “Stateflow®” software support graphical modeling with time-based block diagrams and event-based state machines. “Embedded Coder” supports code generation for embedded systems.

If generated C or C++ code is being deployed in safety-related applications, modeling and code generation are to be complemented by measures and techniques to verify and validate the model and the generated C or C++ code. Applying these measures and techniques in an application-specific manner serves the purpose of translation validation.[3] A successful translation validation provides a high degree of confidence that — for the design instance under consideration — the output of the code generator, compiler, and linker tool chain exhibits equivalent input-output behavior as the model used for production code generation.

The workflow presented in the *Embedded Coder Reference Workflow* describes a translation validation process intended to comply with applicable requirements of the overall software safety lifecycle defined by ISO 26262-6, IEC 61508-3, and EN 50128 respectively, as they relate to verification and validation of models and generated code. The workflow addresses risk levels ASIL A – ASIL D according to ISO 26262, SIL 1 – SIL 3 according to IEC 61508, and SIL 0 – SIL 4 according to EN 50128.

Completing the verification and validation workflow is considered to be equivalent to the use of a certified code generation tool chain consisting of a code generator, compiler, and linker to develop the application under consideration (IEC 61508-3 clause 7.4.4.3).

3. A. Pnueli, M. Siegel, E. Singerman: Translation Validation. Lecture Notes in Computer Science, Vol. 1384, pp. 151-166. Springer, 1998

To fulfill objectives of ISO 26262-6, IEC 61508-3, or EN 50128 related to software development processes, verifying and validating the application software under development (translation validation) is required regardless of the tool chain you use.

The workflow for application-specific verification and validation of models and generated C and C++ code outlined in the *Embedded Coder Reference Workflow* divides the translation validation process into two steps:

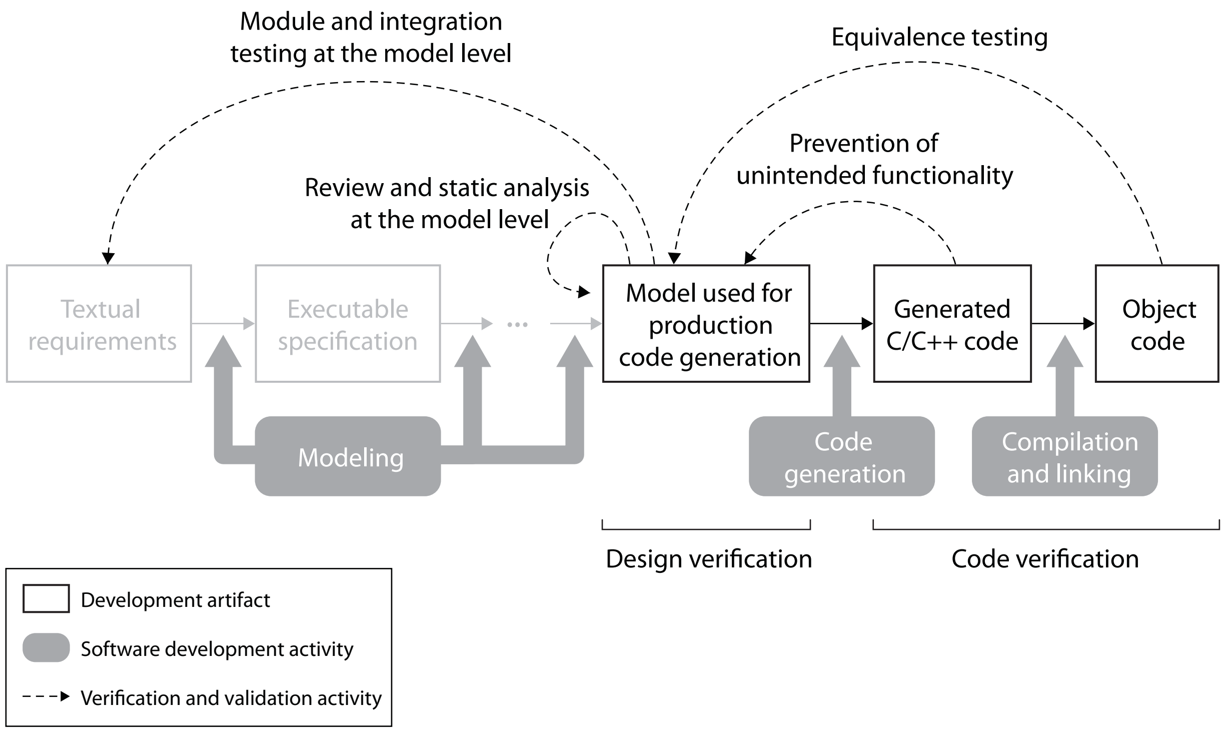
1. Design verification: Demonstrate that the model used for production code generation behaves as specified in its requirements.

2. Code verification: Demonstrate equivalence between the model and the corresponding object code.

The first step combines suitable verification and validation techniques at the model level. The second step mainly relies on behavioral and structural comparison between the model and the generated code.

This two-step approach allows you to complete verification and validation activities, for the most part, at the model level. You can take advantage of the fact that you can reuse model-level tests that are required to verify the generated code.

The following figure shows the suggested translation validation workflow. This workflow is concerned with verifying and validating the model used for production code generation, the generated source code, and the executable object code.


Module and integration testing at the model level
Equivalence testing
Prevention of unintended functionality
Review and static analysis at the model level
Textual requirements
Executable specification
...
Model used for production code generation
Generated C/C++ code
Object code
Modeling
Code generation
Compilation and linking
Design verification
Code verification
Development artifact
Software development activity
Verification and validation activity

Overview of the Workflow for Application-Specific Verification and Validation of Models and Generated C and C++ Code[4]

Open the Artifacts Explorer to access the *Embedded Coder Reference Workflow*. Alternatively, on the MATLAB command line, type `certkitiec`. The *Embedded Coder Reference Workflow* file `certkitiec_ecoder_workflow.pdf` is located in **Embedded Coder > r2013a**.

4. Other development artifacts are grayed out.

Simulink PLC Coder Reference Workflow Overview

The *Simulink PLC Coder Reference Workflow* describes a workflow for application-specific verification and validation of models and structured text code developed using Model-Based Design with PLC code generation. Users of the Simulink PLC Coder software shall carry out this workflow as part of the overall IEC 61508 or IEC 61511 safety lifecycle. Model-Based Design enables automatic generation of IEC 61131-3 structured text code from executable graphical models that can be deployed onto Programmable Logic Controllers (PLCs). Simulink products from MathWorks have become an accepted standard for Model-Based Design. “Simulink” and “Stateflow” software support graphical modeling with time-based block diagrams and event-based state machines. The MATLAB Function block allows including MATLAB algorithms in Simulink models.“Simulink PLC Coder” software supports the generation of IEC 61131 compliant Structured Text code based on Simulink models and Stateflow charts.

If generated structured text is being deployed in safety-related applications, modeling and PLC code generation are to be complemented by measures and techniques to verify and validate the model and the generated PLC code. Applying these measures and techniques in an application-specific manner serves the purpose of translation validation.[5] A successful translation validation provides a high degree of confidence that — for the design instance under consideration — the output of the code generator and programmable logic controller integrated development environment (PLC IDE) tool chain exhibits equivalent input-output behavior as the model used for PLC code generation.

The workflow presented in the *Simulink PLC Coder Reference Workflow* describes a translation validation process intended to comply with applicable requirements of the overall safety lifecycle defined by IEC 61508-3 and IEC 61511, as they relate to verification and validation of models and generated structured text code. The workflow addresses risk levels SIL 1 – SIL 3 according to IEC 61511.

5. A. Pnueli, M. Siegel, E. Singerman: Translation Validation. Lecture Notes in Computer Science, Vol. 1384, pp. 151-166. Springer, 1998

Completing the verification and validation workflow is considered to be equivalent to the use of a certified PLC code generation tool chain to develop the application under consideration (IEC 61508-3, clause 7.4.4.3).

Fulfilling the objectives of IEC 61508-3 and IEC 61511-1 related to software development processes requires verifying and validating the PLC application software under development (translation validation).

The workflow for application-specific verification and validation of models and generated PLC code outlined in the *Simulink PLC Coder Reference Workflow* divides the translation validation process into two steps:

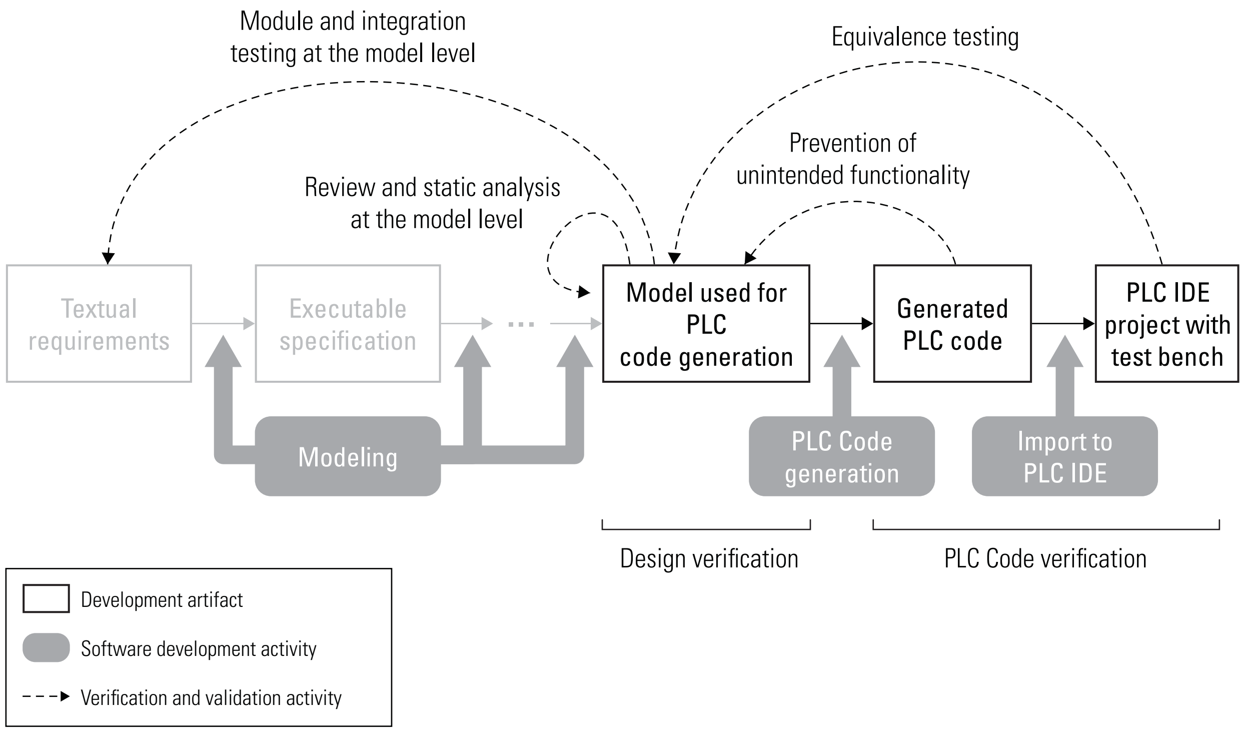
1. Design verification: Demonstrate that the model used for production code generation behaves as specified in its requirements.

2. PLC code verification: Demonstrate equivalence between the model and the generated PLC code.

The first step combines suitable verification and validation techniques at the model level. The second step mainly relies on comparing the model and the generated PLC code.

This two-step approach allows you to complete verification and validation activities, for the most part, at the model level. You can take advantage of the fact that you can reuse model-level tests that when verifying the generated PLC code.

The following figure shows the suggested translation validation workflow. This workflow is concerned with verifying and validating the model used for PLC code generation and the generated PLC code.


Module and integration testing at the model level
Equivalence testing
Prevention of unintended functionality
Review and static analysis at the model level
Textual requirements
Executable specification
Model used for PLC code generation
Generated PLC code
PLC IDE project with test bench
Modeling
PLC Code generation
Import to PLC IDE
Design verification
PLC Code verification
Development artifact
Software development activity
Verification and validation activity

Overview of the Workflow for Application-Specific Verification and Validation of Models and Generated PLC Code[6]

Open the Artifacts Explorer to access the *Simulink PLC Coder Reference Workflow*. Alternatively, on the MATLAB command line, type `certkitiec`. The *Simulink PLC Coder Reference Workflow* file `certkitiec_plccoder_workflow.pdf` is located in **Simulink PLC Coder > r2013a**.

6. Other development artifacts are grayed out.

Simulink Design Verifier Reference Workflow Overview

“Simulink Design Verifier” allows users to generate test cases for Simulink models. The generated test cases provide simulation inputs that exercise functionality captured in the model structure and specified by the test objectives. The test cases, together with the test objectives, are used to verify the model or code running in software-in-the-loop (SIL) or processor-in-the-loop (PIL) modes.

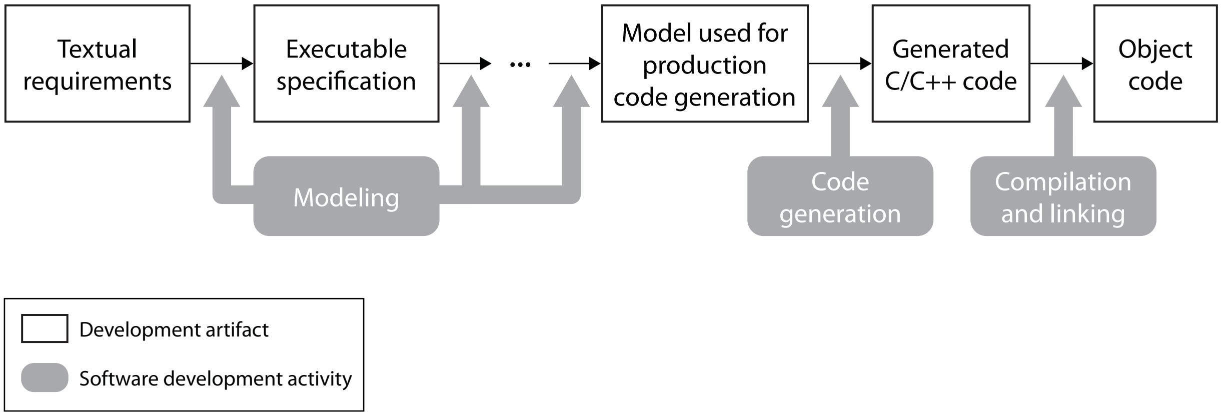
The *Simulink Design Verifier Reference Workflow* provides a reference workflow for Simulink Design Verifier. In particular, it describes how to:

- Leverage the test case generation capability of Simulink Design Verifier in a Model-Based Design process.
- Facilitate seamless functioning of the test case generation capability of the Simulink Design Verifier tool.
- Assess the completeness and adequacy of the generated test cases.

Users of the Simulink Design Verifier software seeking to leverage the certification or qualification credit afforded by the IEC Certification Kit shall carry out this workflow as part of the overall ISO 26262, IEC 61508, or EN 50128 software safety lifecycle.

The *Simulink Design Verifier Reference Workflow* describes use cases for the test case generation capability of Simulink Design Verifier as part of a Model-Based Design process.

During the development of embedded application software, you can use graphical modeling with “Simulink”, “Fixed-Point Designer”, and “Stateflow” to conceptualize the functionality to be implemented. Using this modeling paradigm, the application software to be developed is modeled using time-based block diagrams and event-based state machines. Such a model of the application software is simulated (executed) within the Simulink environment. The model serves as the primary representation of the application software throughout the development process, specifying functionality and design information, and serving as a source for automated code generation with “Embedded Coder”. In practice, this model elaboration is characterized by a step-wise transformation of the application software model from an early executable specification into a model suitable for production code generation, and then finally into production-quality C or C++ code. To accomplish the transformation, the model is enhanced by adding design information and implementation details. The development process becomes the successive refinement of models, followed by automatic code generation and compilation and linking, as shown in the following figure. For details about this development process, see "Embedded Coder Reference Workflow Overview" on page 2-2.



Model-Based Design Process[7]

You can use the test case generation capability provided by the Simulink Design Verifier to generate test cases for the executable specification, the model used for production code generation, or any other interim model created during the modeling phase. You can use the generated test cases to stimulate the executable specification or another stage of the model-based design process.

Open the Artifacts Explorer to access the *Simulink Design Verifier Reference Workflow*. Alternatively, on the MATLAB command line, type `certkitiec`. The *Simulink Design Verifier Reference Workflow* file `certkitiec_sldv_workflow.pdf` is located in **Simulink Design Verifier > r2013a**.

7. Solid arrows in the figure indicate the succession of software development activities.

Simulink Verification and Validation Reference Workflow Overview

“Simulink Verification and Validation” allows users to:

- Check Simulink and Stateflow models for compliance with design and coding guidelines.
- Identify untested portions of models using structural coverage metrics.

The *Simulink Verification and Validation Reference Workflow* provides a reference workflow for Simulink Verification and Validation. In particular, it describes how to:

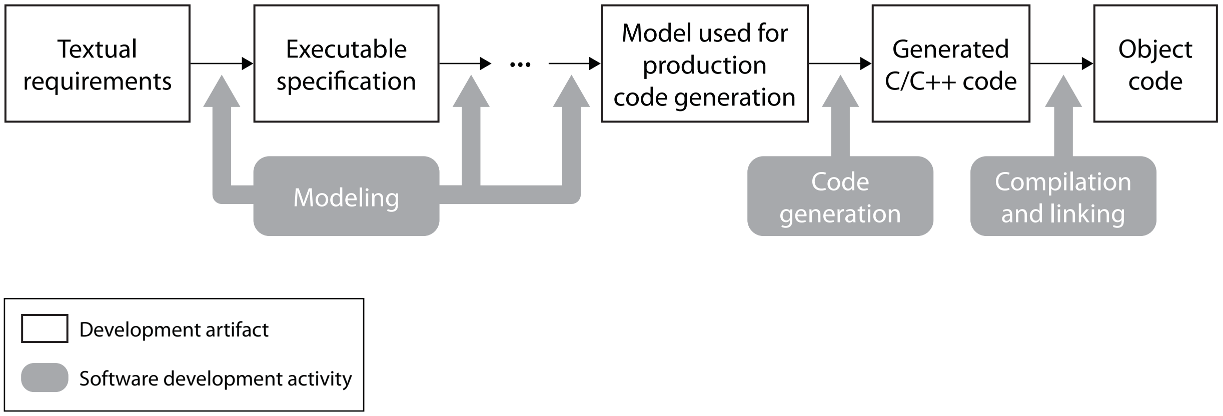
- Leverage the model compliance checking and model coverage analysis capabilities of Simulink Verification and Validation in a Model-Based Design process.
- Check these capabilities are functioning as expected.

Users of the Simulink Verification and Validation tool seeking to leverage the certification or qualification of the tool shall carry out this workflow as part of the overall ISO 26262, IEC 61508, or EN 50128 software safety lifecycle.

The *Simulink Verification and Validation Reference Workflow* describes use cases for the following capabilities of Simulink Verification and Validation as part of a Model-Based Design process:

- Model compliance checking
- Model coverage analysis

During the development of embedded application software, you can use graphical modeling with “Simulink”, “Fixed-Point Designer”, and “Stateflow” to conceptualize the functionality to be implemented. Using this modeling paradigm, the application software to be developed is modeled using time-based block diagrams and event-based state machines. Such a model of the application software is simulated (executed) within the Simulink environment. The model serves as the primary representation of the application software throughout the development process, specifying functionality and design information, and serving as a source for automated code generation with Embedded Coder. In practice, this model elaboration is characterized by a step-wise transformation of the application software model from an early executable specification into a model suitable for production code generation, and then finally into C or C++ code. To accomplish the transformation, the model is enhanced by adding design information and implementation details. The development process becomes the successive refinement of models, followed by automatic code generation and compilation and linking, as shown in the following figure. For details about this development process, see "Embedded Coder Reference Workflow Overview" on page 2-2.



Model-Based Design Process[8]

You can use the model compliance checking and model coverage analysis capabilities of Simulink Verification and Validation to verify or validate the executable specification, the model used for production code generation, or any other interim model created during the modeling phase.

Open the Artifacts Explorer to access the *Simulink Verification and Validation Reference Workflow*. Alternatively, on the MATLAB command line, type `certkitiec`. The *Simulink Verification and Validation Reference Workflow* file `certkitiec_slvnv_workflow.pdf` is located in **Simulink Verification and Validation > r2013a**.

8. Solid arrows in the figure indicate the succession of software development activities.

Polyspace Client/Server for C/C++ Reference Workflow Overview

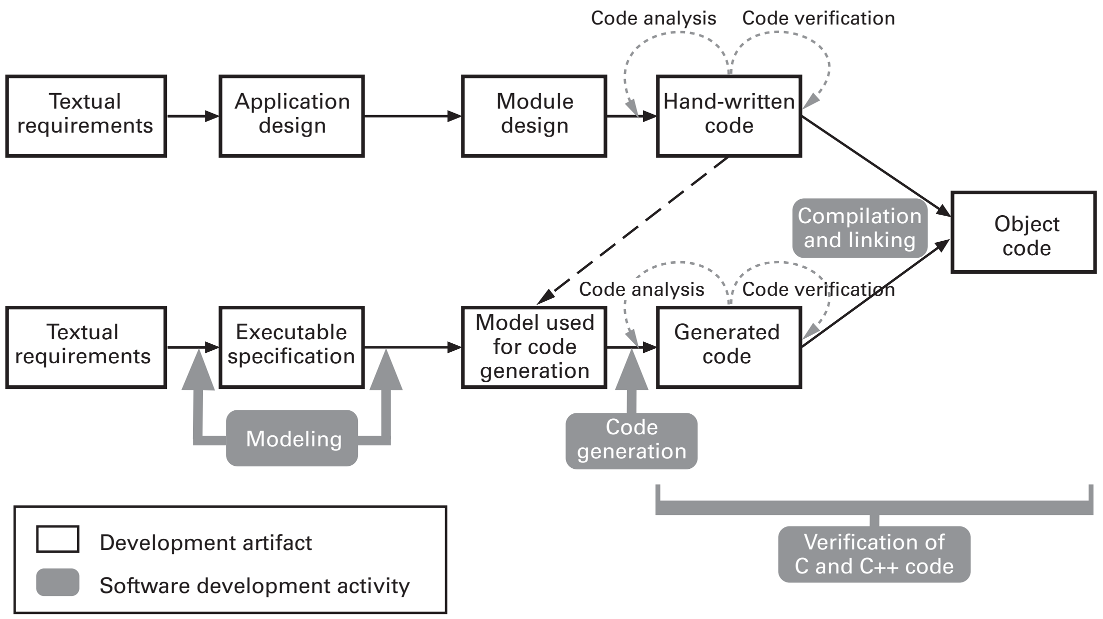
The *Polyspace Client/Server for C/C++ Reference Workflow* describes a reference workflow for application-specific verification of C or C++ source code for embedded application software. It addresses hand-written, automatically generated, and mixed code. This workflow is valid whether you are developing code or auditing code received from others.

You can use Polyspace products to carry out this workflow as part of the overall IEC 61508, ISO 26262, or EN 50128 software safety lifecycle.

If C or C++ code is being deployed in safety-related applications, the source code shall be verified. The *Polyspace Client/Server for C/C++ Reference Workflow* describes a code verification process intended to comply with applicable requirements of the overall software safety lifecycles defined by IEC 61508-3 , ISO 26262, and EN 50128 respectively, as they relate to verification of hand-written, generated, or mixed code. The workflow addresses risk levels ASIL A – ASIL D according to ISO 26262, SIL 1 – SIL 3 according to IEC 61508, and SIL 0 - SIL 4 according to EN 50128.

To fulfill the objectives of ISO 26262-6, IEC 61508-3, and EN 50128 related to the software development process, verification of the application software under development is required, regardless of the tool chain used.

The workflow for application-specific verification is different for hand-written code, generated code, and mixed code. The following figure shows the suggested verification workflow for hand-written and mixed code. For generated code, this workflow can also be used to provide added assurance to the one described in *Embedded Coder Reference Workflow*.



Workflow for Application-Specific Verification of C and C++ Code using Polyspace® Products

Polyspace products can perform code verification as part of either variant of this workflow.

The reference workflow outlined in the *Polyspace Client/Server for C/C++ Reference Workflow* comprises the following activities:

1. **Definition of software quality objectives** – Define quality levels for each software module.
2. **Code analysis** – Demonstrate compliance with C or C++ coding standards
3. **Code verification** – Prove the absence of run-time errors in the code, and confirm that the results meet your software quality objectives.

The first activity in this process sets the quality objectives for your application, while the other activities use Polyspace products to demonstrate software quality in relation to the metrics established.

Open the Artifacts Explorer to access the *Polyspace Client/Server for C/C++ Reference Workflow*. Alternatively, on the MATLAB command line, type `certkitiec`. The *Polyspace Client/Server for C/C++ Reference Workflow* file `certkitiec_polyspace_workflow.pdf` is located in **Polyspace Client/Server for C/C++ > r2013a**.

Certification Process

- "Define Certification Objectives and Requirements" on page 3-2
- "Certify or Qualify Software Tools" on page 3-3
- "Document Evidence of Using Tools Within Referenced Workflows" on page 3-4

Define Certification Objectives and Requirements

Before using the IEC Certification Kit product, define your certification objectives and requirements.

- Identify the scope of your certification activities, such as the application to certify.
- Decide on the applicable safety standards and the required Safety Integrity Level (SIL) or Automotive Safety Integrity Level (ASIL).
- Determine the software development processes and software tool chain to use.
- Define tool certification or qualification requirements, including the tools and versions to certify or qualify.

Certify or Qualify Software Tools

The ISO 26262, IEC 61508, and EN 50128 standards include requirements or recommendations to use certified or qualified tools. You can use tool certification or prequalification evidence from the IEC Certification Kit product to document compliance with the requirements or recommendations concerning tool certification or qualification.

Note Using certified or qualified tools does not ensure the safety of the application under development.

The IEC Certification Kit product provides tool certification and prequalification evidence for the following MathWorks products:

- Embedded Coder
- Simulink PLC Coder
- Simulink Design Verifier
- Simulink Verification and Validation
- Polyspace Client for C/C++; Polyspace Server for C/C++

The IEC Certification Kit product follows an in-context approach to tool certification and qualification. This approach is based on specific workflows to be used when applying the certified and qualified tools to develop or verify software for ISO 26262, IEC 61508, and EN 50128 applications. The applicant must ensure that the tools are used within the referenced workflows and constraints specified in the certificates.

Document Evidence of Using Tools Within Referenced Workflows

Use IEC Certification Kit artifact templates to document evidence of using MathWorks tools within referenced workflows and with the constraints specified in the corresponding certificate. The documentation activities for each tool can include the following:

- Customize and complete the *Conformance Demonstration Template* provided for the tool.
- For ISO 26262 tool qualification, review the *ISO 26262 Tool Qualification Package* template provided for the tool for applicability to the application under consideration, and tailor and complete the information.

In this section...
"ISO 26262 Tool Qualification Artifacts" on page 3-4
"IEC 61508 Tool Certification Artifacts" on page 3-5
"EN 50128 Tool Certification Artifacts" on page 3-8

ISO 26262 Tool Qualification Artifacts

The IEC Certification Kit product provides support for creating ISO 26262 tool qualification artifacts for the following products:

- Embedded Coder
- Simulink Design Verifier
- Simulink Verification and Validation
- Polyspace Client for C/C++; Polyspace Server for C/C++

For details, see the ISO 26262 Tool Qualification Package documents for these products.

Note Some safety standards, including IEC 61508, do not have a formal concept of certification credits. The amount of credit for the use of certified or qualified tools is dependent on the applicant's development, verification and validation processes, and how the applicant uses the tools within those processes. The applicant should propose and discuss an initial version of the compliance package, including tool qualification data, to the certification authority or internal assessor early in the development lifecycle.

IEC 61508 Tool Certification Artifacts

The IEC Certification Kit product provides support for creating the following artifacts related to tool certification according to IEC 61508.

Products	Purpose	References	Artifacts and Documents[1]
Embedded Coder	Tool certification evidence for code generator	• IEC 61508-3 Clause 7.4.4 • IEC 61508-3 Table A-3 (4a) "Certified tools and certified translators"	• Certificate Z10 11 12 67052 014 • Certificate report MN72051C
	Documentation of reference workflow	N/A	*Embedded Coder Reference Workflow*
	Evidence for using the code generator within the referenced workflows and within the constraints specified in its certificate	N/A	Customized and completed *Conformance Demonstration Template*

Products	Purpose	References	Artifacts and Documents[1]
Simulink PLC Coder	Tool certification evidence for code generator	• IEC 61508-3 Clause 7.4.4 • IEC 61508-3 Table A-3 (4a) "Certified tools and certified translators"	• Certificate Z10 11 01 67052 007 • Certificate report MN76171C
	Documentation of reference workflow	N/A	*Simulink PLC Coder Reference Workflow*
	Evidence for using the code generator within the referenced workflows and within the constraints specified in its certificate	N/A	Customized and completed *Conformance Demonstration Template*
Simulink Design Verifier	Tool certification evidence for model verification tool	• IEC 61508-3 Clause 7.4.4 • IEC 61508-3 Table A-3 (4a) "Certified tools and certified translators"	• Certificate Z10 11 12 67052 013 • Certificate report MN83534C
	Documentation of reference workflow	N/A	*Simulink Design Verifier Reference Workflow*
	Evidence for using the verification tool within the referenced workflows and within the constraints specified in its certificate	N/A	Customized and completed *Conformance Demonstration Template*

Products	**Purpose**	**References**	**Artifacts and Documents**[1]
Simulink Verification and Validation	Tool certification evidence for model verification tool	• IEC 61508-3 Clause 7.4.4 • IEC 61508-3 Table A-3 (4a) "Certified tools and certified translators"	• Certificate Z10 11 12 67052 013 • Certificate report MN83534C
	Documentation of reference workflow	N/A	*Simulink Verification and Validation Reference Workflow*
	Evidence for using the verification tool within the referenced workflows and within the constraints specified in its certificate	N/A	Customized and completed *Conformance Demonstration Template*
Polyspace Client for C/C++; Polyspace Server for C/C++	Tool certification evidence for code verification tool	• IEC 61508-3 Clause 7.4.4 • IEC 61508-3 Table A-3 (4a) "Certified tools and certified translators"	• Certificate Z10 11 12 67052 015 • Certificate Report MN74651C
	Documentation of reference workflow	N/A	*Polyspace Client/Server for C/C++ Reference Workflow*
	Evidence for using the verification tool within the referenced workflows and within the constraints	N/A	Customized and completed *Conformance*

Products	Purpose	References	Artifacts and Documents[1]
	specified in its certificate		*Demonstration Template*

[1]For file names and locations, see "IEC Certification Kit Component Summary" on page 1-4.

EN 50128 Tool Certification Artifacts

The IEC Certification Kit product provides support for creating EN 50128 tool qualification artifacts for the following products:

- Embedded Coder
- Simulink Design Verifier
- Simulink Verification and Validation
- Polyspace Client for C/C++; Polyspace Server for C/C++

Validate Software Tools

- “Software Tool Validation” on page 4-2
- “Run Test Cases and Procedures for Embedded Coder” on page 4-3
- “Run Test Cases and Procedures for Simulink® Verification and Validation™” on page 4-4
- “Run Test Cases and Procedures for Polyspace Client/Server for C/C++” on page 4-6

Software Tool Validation

Some safety standards recommend the validation of software tools, using application-independent test cases to:

- Demonstrate that a software tool complies with its specified requirements.
- Examine the reaction of the software tool to anomalous operating conditions.

The IEC Certification Kit product provides exemplary test cases and test procedures that you can use to automate tool validation tests for the following products:

- Embedded Coder
- Simulink Verification and Validation (Model Coverage Analysis, Model Compliance Checking)
- Polyspace Client/Server for C/C++

The exemplary test cases provided with the IEC Certification Kit product are templates that you can modify and extend to create test suites that cover the requirements that are relevant for your application, your specific tool configuration, operating environment, and so on.

Note MathWorks acknowledges the Automotive Code Validation Suite (AVS) as the initial test suite used with Embedded Coder, as described in the following article:

```
http://www.mathworks.com/company/pressroom/article31185.html
```

Run Test Cases and Procedures for Embedded Coder

To execute the test cases and procedures for Embedded Coder:

1. Copy the `*matlabroot*/toolbox/qualkits/iec/ecoder/r2013a` folder and its subfolders to a location to which you have write access. Use that location to run the test cases and procedures.

   Note

   - To execute the test procedure, you must have an Embedded Coder license.
   - Some test models require Stateflow and Simulink licenses.

2. Open the file `certkitiec_ecoder_modelList.m` in the relocated folder.
3. Edit the file to specify the test cases (that is, test models and supporting files) that you want to execute. Check that the models and files that you specify exist in their specified locations in the `/tests` subfolder.
4. Optionally, edit the file to specify baselines corresponding to the tests. Check that the baselines that you specify exist in the `/baselines` subfolder.
5. Save the file.
6. To run the tests and generate a validation report, execute the file `certkitiec_ecoder_tests.m`. You can invoke it from the MATLAB command line or in the Certification Artifacts Explorer. Test reports are generated in HTML format and are placed in the `outputs` subfolder.
7. Confirm that the test reports are generated without errors or warnings.
8. Review the generated test reports for expected results.

Run Test Cases and Procedures for Simulink Verification and Validation

To execute the test cases and procedures for Simulink Verification and Validation (Model Coverage Analysis, Model Compliance Checking):

1. Copy the `*matlabroot*/toolbox/qualkits/iec/slvnv/r2013a` folder and its subfolders to a location to which you have write access. Use that location to run the test cases and procedures.

   Note

   - To run the tests and generate reports, you must have MATLAB Report Generator™ and Simulink Report Generator licenses.
   - Some model coverage RPT files require Fixed-Point Designer, Stateflow, and Simulink Design Verifier licenses.

2. Open the files `certkitiec_slvnv_tests*.xls` in the relocated folder.
3. Edit the files to specify the test cases (that is, test models and supporting files) that you want to execute, the expected results, and additional information. Check that the models and files that you specify exist in their specified locations in the `/tests` subfolder.
4. Save the files.
5. To run the tests and generate reports, execute the files `certkitiec_slvnv_tests*.rpt`. You can invoke them in the Certification Artifacts Explorer, from the MATLAB command line, or from the Report Explorer, as follows:

   - In the Certification Artifacts Explorer, right-click an RPT file and select **Execute Tests**.
   - At the MATLAB command line, enter the command

     ```
     report ('rpt_file')
     ```

     where *`rpt_file`* is the name of the test procedure.

- To open Report Explorer, double-click an RPT file, or in Certification Artifacts Explorer, right-click an RPT file and select **Open File**. In Report Explorer, select **File > Report**.

Simulink Report Generator creates the test reports and places them in the `outputs` subfolder.

Note

- Before you execute model coverage RPT files, set the Java® heap size for your MATLAB session to at least 512 MB. To check the Java heap size, open the MATLAB Preferences dialog box and select **General > Java Heap Memory**. If the **Java Heap Size** value is less than 512 MB, change it to 512 MB, click **OK**, and restart MATLAB. (If the maximum available heap size value is less than 512 MB, select the maximum value.) This may help you avoid `java.lang.OutOfMemoryError` messages.
- Before you execute each model coverage RPT file, start a new MATLAB session.

---

6. Confirm that the test reports are generated without errors or warnings.
7. Review the generated test reports for expected results. The tool validation report for the ISO 26262 Model Advisor checks provides the expected and actual results for the overall check and subchecks. If one of the subchecks warns, the overall check result is a warning.

Run Test Cases and Procedures for Polyspace Client/Server for C/C++

To execute test cases and procedures for Polyspace Client/Server for C/C++:

1. On the Polyspace server machine, copy the `*matlabroot*/toolbox/qualkits/iec/polyspace/r2013a` folder and its subfolders to a location to which you have write access. Use that location to run the test cases and procedures.
2. From the top level of the relocated folder, `cd` into the subfolder `tests`.
3. To run the tests and generate reports:
   - On a Windows® system, use the command `certkitiec_polyspace_tests.bat`.
   - On a Linux® system, make sure that Perl and Polyspace are in the current PATH and use the command `certkitiec_polyspace_tests.sh`.
4. As the tests run, reports are generated in the `outputs` subfolder of the relocated folder.
5. Confirm that the test reports are generated without errors or warnings.
6. Review the generated test reports for expected results.

For examples of generated reports, see the `outputs` subfolder of the relocated folder:

- `certkitiec_polyspace_qualificationreport_code_metrics.txt`
- `certkitiec_polyspace_qualificationreport_misrac.txt`
- `certkitiec_polyspace_qualificationreport_misracpp.txt`
- `certkitiec_polyspace_qualificationreport_tor.txt`

Access and Manage Certification Artifacts

- “Access Artifacts Using the Certification Artifacts Explorer” on page 5-2
- “Manage Artifacts Using the Certification Artifacts Explorer” on page 5-5
- “Delete Certification Packages from the Certification Artifacts Explorer” on page 5-7
- “Certification Artifacts Explorer Limitations” on page 5-8

Access Artifacts Using the Certification Artifacts Explorer

In this section...

"Certification Artifacts in the IEC Certification Kit Product" on page 5-2
"What Is a Certification Package?" on page 5-2
"How To Access Certification Artifacts" on page 5-3

Certification Artifacts in the IEC Certification Kit Product

The IEC Certification Kit product includes the following certification artifacts:

- Certification and qualification evidence
- Documents and templates

For more information about the certification artifacts that are part of the IEC Certification Kit product, see "IEC Certification Kit Component Summary" on page 1-4.

For more information about certifying or qualifying software tools, see "Process for Standard Compliance or Certification".

What Is a Certification Package?

A certification package is a group of certification artifacts that you use to certify your project. The Certification Artifacts Explorer displays:

- The certification artifacts that are part of the IEC Certification Kit product.
- Certification packages that you create.

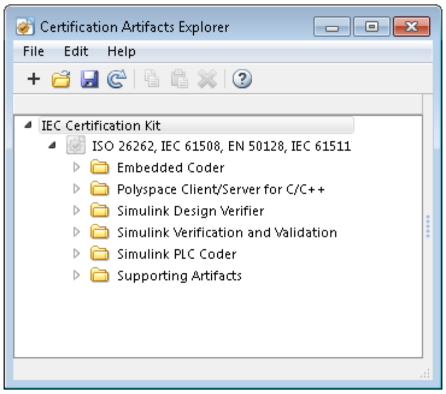
How To Access Certification Artifacts

You can use the Certification Artifacts Explorer to access certification artifacts. To start the Certification Artifacts Explorer, use one of the following methods.

To start the Certification Artifacts Explorer...	Do this:
From the MATLAB toolstrip	**1** Install the **Compliance Artifacts Explorer** App: • Click the **Apps** tab and select **Install App**. • Navigate to `*matlabroot*/toolbox/qualkits/iec/` and open `Compliance Artifacts Explorer.mlappinstall`. **2** Click the **Apps** tab and select **Compliance Artifacts Explorer**.
From the MATLAB command line	Enter `certkitiec`.

The Certification Artifacts Explorer window displays the certification artifacts that are available with the IEC Certification Kit product. If the IEC Certification Kit product contains artifacts for more than one release, the Certification Artifacts Explorer lists the artifacts for each release. As you select folders and files, relevant information about the current selection is dynamically displayed in the status bar. Additionally,

- To display the properties of a certification package, right-click the package name and select **Properties**.
- To open an artifact, right-click the artifact and select **Open File**.
- Other right-click actions might be available depending on the type or state of an artifact, including **Copy**, **Paste**, **Delete**, **Open Folder**, and **Generate Traceability Matrix**.



IEC Certification Kit
ISO 26262, IEC 61508, EN 50128, IEC 61511
Embedded Coder
Polyspace Client/Server for C/C++
Simulink Design Verifier
Simulink Verification and Validation
Simulink PLC Coder
Supporting Artifacts

Manage Artifacts Using the Certification Artifacts Explorer

To manage certification artifacts using the Certification Artifacts Explorer:

1 Create a new certification package.

Tip Most certification package actions can be initiated in multiple ways. For example, to create a new package, you can:

- Select the menu item **File > New**.
- Click the toolbar icon **Create new certification package**.
- Right-click an existing package and select **New**.

---

2 Name the certification package.

3 Define the location where the Certification Artifacts Explorer stores the certification package (use right-click **> Properties**).

4 Save the certification package. The saved package has a KIT extension.

5 Copy the certification artifacts for the product of interest into the certification package.

6 Delete certification artifacts that are not required for your project.

7 Optionally, add related files to the certification package using a file browser such as Microsoft® Windows Explorer.

Tip When you add files, to refresh the file list, use **File > Refresh**.

---

8 Use the Certification Artifacts Explorer to access certification artifacts. For a list of artifacts that you might need to access and modify, see “Certify or Qualify Software Tools” on page 3-3.

When you create and save new certification packages, the Certification Artifacts Explorer displays them. The certification packages that are listed remain visible unless you delete them from the Certification Artifacts Explorer.

Delete Certification Packages from the Certification Artifacts Explorer

The Certification Artifacts Explorer displays certification packages that you create or open. If you delete a certification package from the Certification Artifacts Explorer, the files associated with the package are still available on your computer. To delete the files, use a file browser such as Windows Explorer.

Certification Artifacts Explorer Limitations

The Certification Artifacts Explorer has the following limitation:

- For optimal performance, Microsoft Internet Explorer® must be available on your machine. Internet Explorer does not have to be your default web browser.

Support Certification-Related Development Activities

- "Generate a Traceability Matrix" on page 6-2
- "Add Comments to a Traceability Matrix" on page 6-6
- "Traceability Matrix Limitations" on page 6-8
- "Display Bug Reports" on page 6-9

Generate a Traceability Matrix

In this section...

"About Traceability Matrices" on page 6-2

"Prerequisites for Generating a Traceability Matrix" on page 6-3

"How to Generate a Traceability Matrix" on page 6-4

About Traceability Matrices

When you use Model-Based Design and production code generation to develop application software components, you can generate a *traceability matrix*. The traceability matrix provides traceability among model objects, generated code, and model requirements. You can add comments to the generated traceability matrix. If you change the model and regenerate the traceability matrix, the software retains your comments.

For a given model, the generated traceability matrix can provide information about:

- Model objects that are traceable between the model and generated code, such as Simulink blocks, Stateflow objects, and MATLAB functions.
- Model objects that are untraceable between the model and generated code, such as eliminated and virtual blocks.
- Requirements documents that you link to model objects using the Simulink Verification and Validation Requirements Management Interface (RMI).

Generate the traceability matrix using either the `iec.ExportTraceReport` function from the MATLAB Command Window or the **Generate Traceability Matrix** button in the generated HTML code generation report for your model. Either method creates an XLS file that contains the following worksheets:

- **Model Information** — Summary of the model configuration and checksum. The summary includes the model name, version, author, creation date, last saved by, last updated date, checksum, and the selection of **Traceability Report Contents** parameters.

- **Code Interface** — Information about the generated code interface, such as function prototype and timing information for the model initialize and step functions.
- **Code Files** — File folders and names of the generated code files.
- **Report** — Traceability information for each model object, including model, generated code, and requirements. Each row in the worksheet pertains to a single occurrence of a model object. The information for a model object is in more than one row if the object:
  - Appears more than once in the generated code.
  - Links to more than one requirement.

Prerequisites for Generating a Traceability Matrix

Before generating a traceability matrix for model objects, generated code, and model requirements, perform the following steps:

1. Optionally, attach requirements documents. For more information, see "Requirements Traceability" in the Simulink Verification and Validation documentation.

2. In the Configuration Parameters dialog box, on the **Code Generation > Report** pane, select:

   a. **"Create code generation report"**

   b. At least one of the following **Traceability Report Contents** parameters:

      - **"Eliminated / virtual blocks"**
      - **"Traceable Simulink blocks"**
      - **"Traceable Stateflow objects"**
      - **"Traceable MATLAB functions"**

   Tip If you want to generate the traceability matrix directly from the code generation report, select **"Open report automatically"**.

   ---

3. Generate code for the model.

Tip You do not have to build an executable to generate a traceability matrix. To generate code only, on the **Code Generation > General** pane, select **Generate code only**.

How to Generate a Traceability Matrix

To generate a traceability matrix:

1 Open the model if it is not already open.

2 Check that you have completed the "Prerequisites for Generating a Traceability Matrix" on page 6-3.

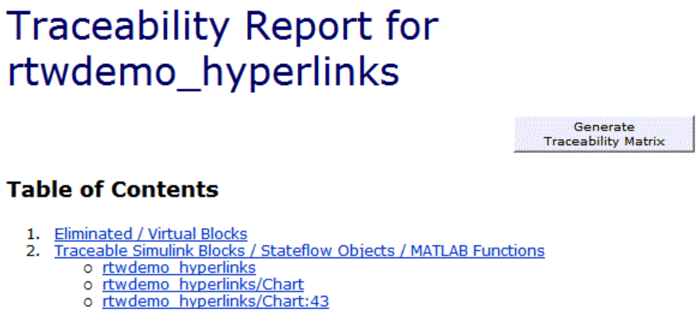
3 Generate the traceability matrix using one of the following methods:

- In the MATLAB Command Window, enter the following command, where *model_name* is the name of the model:

  ```
  iec.ExportTraceReport('model_name')
  ```

  The software generates the traceability matrix.

- Open the code generation report for the model if it is not already open. Go to the **Traceability Report** section and click the **Generate Traceability Matrix** button. For example:

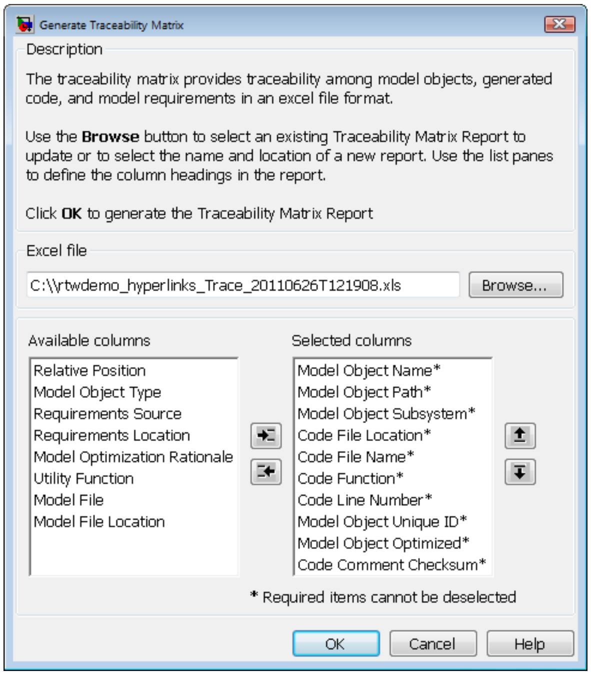


Traceability Report for rtwdemo_hyperlinks

Table of Contents

1. Eliminated / Virtual Blocks
2. Traceable Simulink Blocks / Stateflow Objects / MATLAB Functions
   - rtwdemo_hyperlinks
   - rtwdemo_hyperlinks/Chart
   - rtwdemo_hyperlinks/Chart:43

When you click the button, the Generate Traceability Matrix dialog box appears.



The traceability matrix provides traceability among model objects, generated code, and model requirements in an excel file format.

Use the **Browse** button to select an existing Traceability Matrix Report to update or to select the name and location of a new report. Use the list panes to define the column headings in the report.

Click **OK** to generate the Traceability Matrix Report

Excel file: C:\\rtwdemo_hyperlinks_Trace_20110626T121908.xls

Available columns	Selected columns
Relative Position	Model Object Name*
Model Object Type	Model Object Path*
Requirements Source	Model Object Subsystem*
Requirements Location	Code File Location*
Model Optimization Rationale	Code File Name*
Utility Function	Code Function*
Model File	Code Line Number*
Model File Location	Model Object Unique ID*
	Model Object Optimized*
	Code Comment Checksum*

* Required items cannot be deselected

You can use this dialog box to browse to an existing matrix file to update or specify a new matrix file to create. Optionally, you can also use this dialog box to select and order the columns that appear in the generated matrix. Click **OK** to update or create the specified report.

4 Review the traceability matrix and add comments in new columns. For more information, see “Add Comments to a Traceability Matrix” on page 6-6.

Add Comments to a Traceability Matrix

In this section...
"Requirements for Adding Comments to a Traceability Matrix" on page 6-6
"How To Retain Comments" on page 6-7

Requirements for Adding Comments to a Traceability Matrix

You can add comments to the traceability matrix that you generated using the `iec.ExportTraceReport` function.

To add comments to the traceability matrix, you must:

- Create new columns for your comments.
- Use unique column headings. Columns that you add must have headings.
- Add at least one entry to the column, other than the column heading.
- Retain the following columns:
  - Model Object Name
  - Model Object Path
  - Model Object Subsystem
  - Code File Location
  - Code File Name
  - Code Function
  - Code Line Number
  - Model Object Unique ID
  - Model Object Optimized
  - Code Comment Checksum

Note Comments must resolve to a text string. For example, a link to an image resolves to a text string, but a copy of the image does not.

How To Retain Comments

To regenerate a traceability matrix and retain your comments:

1. Navigate to the working folder of the model.

2. Optionally, regenerate code for your model. Regenerating code before generating the traceability matrix increases the likelihood that you have the latest model-to-code traceability information.

3. In the MATLAB Command Window, enter the following command. *`file_name`* is the name of the existing traceability matrix that you are regenerating. If the existing traceability matrix is in a different folder, include the full path to that folder in *`path`*.

   ```
   iec.ExportTraceReport('model_name', 'file_name', 'path')
   ```

   The traceability matrix regenerates.

Traceability Matrix Limitations

The traceability matrix generation capability has the following limitations:

- Works with the Microsoft Windows platform only.
- Does not support referenced models. When you generate a traceability matrix for a model that contains referenced models, the traceability matrix contains information about the Model block only. The traceability matrix does not contain information about the contents of the referenced model. If your model contains referenced models, generate a traceability matrix for the top-level model and each referenced model separately.
- Does not support models that use the model configuration option **Classic call interface** (`GRTInterface`).
- In most cases, identifies comments that you add to the traceability matrix, but when comments cannot be identified, the traceability matrix includes the text:

  Row is not unique: *`comment`*
- Does not support information stored in external `.req` files. For example, when you generate a traceability matrix for a model with externally stored requirements information, the traceability matrix does not include the requirements information.

Display Bug Reports

The IEC Certification Kit product provides a set of Model Advisor checks to display bug reports for supported MathWorks products. Reports generated by these checks may be used as artifacts in the compliance demonstration process.

See "IEC Certification Kit Checks" on page 8-2 for details on the available checks.

Functions — Alphabetical List

certkitiec

Purpose

Open Certification Artifacts Explorer for IEC Certification Kit

Syntax

`certkitiec`

Description

`certkitiec` opens the Certification Artifacts Explorer and displays certification artifacts.

Tips

- For optimal performance, Microsoft Internet Explorer must be available on your machine. Internet Explorer does not have to be your default web browser.

Alternatives

Launch the Certification Artifact Explorer from the MATLAB toolstrip:

1. Install the **Compliance Artifacts Explorer** App:
   - Click the **Apps** tab and select **Install App**.
   - Navigate to `*matlabroot*/toolbox/qualkits/iec/` and open `Compliance Artifacts Explorer.mlappinstall`.
2. Click the **Apps** tab and select **Compliance Artifacts Explorer**.

Purpose

Generate XLS file that contains traceability matrix

Syntax

```
iec.ExportTraceReport('model_name')
iec.ExportTraceReport('model_name', 'file_name')
iec.ExportTraceReport('model_name', 'file_name', 'path')
```

Description

`iec.ExportTraceReport('`*`model_name`*`')` generates an XLS file that contains a "Traceability Matrix" on page 7-4. *`model_name`* is the name of the model.

`iec.ExportTraceReport('`*`model_name`*`', '`*`file_name`*`')` generates an XLS file that contains a "Traceability Matrix" on page 7-4. *`file_name`* is a string that specifies the name of the file. The first time that you call `iec.ExportTraceReport`, *`file_name`* is optional. If you do not provide *`file_name`*, the function names the file using the following convention. *`modelUpdate`* is the date and time that you last updated the model:

`model_name``_Trace_`*`modelUpdate`*`.xls`

To regenerate the traceability matrix, you must specify *`file_name`*.

`iec.ExportTraceReport('`*`model_name`*`', '`*`file_name`*`', '`*`path`*`')` generates an XLS file that contains a "Traceability Matrix" on page 7-4. *`path`* is an optional string that specifies the full path to the location where you want the software to save the file.

Tips

- The `iec.ExportTraceReport` function works on Microsoft Windows platforms only.
- To include requirements documentation in the traceability matrix, attach requirements documents to the model before using `iec.ExportTraceReport`.
- You must generate a code generation traceability report (requires an Embedded Coder license) for your model before using `iec.ExportTraceReport`.
- The `iec.ExportTraceReport` function does not support referenced models. When you generate a traceability matrix for a model that contains referenced models, the traceability matrix contains

information about the Model block only. The traceability matrix does not contain information about the contents of the referenced model. If your model contains referenced models, generate a traceability matrix for the top-level model and each referenced model separately.

- The `iec.ExportTraceReport` function does not support models that use the model configuration option **Classic call interface** (`GRTInterface`).
- In most cases, the `iec.ExportTraceReport` function identifies comments that you add to the traceability matrix. When the function cannot identify comments, the traceability matrix includes the text:

  Row is not unique: *`comment`*

For more information, see "Prerequisites for Generating a Traceability Matrix" on page 6-3.

Definitions

Traceability Matrix

A traceability matrix provides traceability among model objects, generated code, and model requirements. You can add comments to the generated traceability matrix. If you change the model and regenerate the traceability matrix, the software retains your comments.

Examples

Generate a traceability matrix with traceability between model objects and generated code for the `rtwdemo_hyperlinks` model:

Note This example requires an Embedded Coder license.

```
% Open the model.
open_system('rtwdemo_hyperlinks');
% Generate code only.
set_param('rtwdemo_hyperlinks', 'GenCodeOnly', 'on');
% Initiate the build process.
rtwbuild('rtwdemo_hyperlinks');
% Generate a traceability matrix.
```

```
iec.ExportTraceReport('rtwdemo_hyperlinks');
```

Generate a traceability matrix with traceability among model objects, generated code, and model requirements for the `slvnvdemo_fuelsys_docreq` model:

Note This example requires a Simulink Verification and Validation license.

```
% Open the model.
open_system('slvnvdemo_fuelsys_docreq');
% Select the code generation report and traceability report parameters.
set_param('slvnvdemo_fuelsys_docreq', 'GenerateReport', 'on');
set_param('slvnvdemo_fuelsys_docreq', 'GenerateTraceReport', 'on');
set_param('slvnvdemo_fuelsys_docreq', 'GenerateTraceReportSl', 'on');
set_param('slvnvdemo_fuelsys_docreq', 'GenerateTraceReportSf', 'on');
set_param('slvnvdemo_fuelsys_docreq', 'GenerateTraceReportEml', 'on');
% Generate code only.
set_param('slvnvdemo_fuelsys_docreq', 'GenCodeOnly', 'on');
% Initiate the build process.
rtwbuild('slvnvdemo_fuelsys_docreq');
% Generate a traceability matrix.
iec.ExportTraceReport('slvnvdemo_fuelsys_docreq');
```

Alternatives

You can generate a traceability matrix directly from the code generation report for your model. Go to the **Traceability Report** section and click the **Generate Traceability Matrix** button.

How To

- “Generate a Traceability Matrix” on page 6-2
- “Add Comments to a Traceability Matrix” on page 6-6
- “Code Tracing”
- “Requirements Traceability”

Model Advisor Checks

IEC Certification Kit Checks

In this section...

"Display bug reports using IEC Certification Kit checks" on page 8-3

"Display bug reports for Simulink® Verification and Validation™" on page 8-4

"Display bug reports for Simulink® Design Verifier™" on page 8-5

"Display bug reports for Simulink® PLC Coder™" on page 8-6

"Display bug reports for IEC Certification Kit" on page 8-7

"Display bug reports for Polyspace products for C/C++" on page 8-8

"Display bug reports for Embedded Coder" on page 8-8

Display bug reports using IEC Certification Kit checks

You can use the IEC Certification Kit Model Advisor checks to display bug reports for supported products.

Display bug reports for Simulink Verification and Validation

Display bug reports for the Simulink Verification and Validation R2013a product.

Description

Run this check to display the bug reports for Simulink Verification and Validation R2013a that are available at www.mathworks.com/support/bugreports.

Note This check does not determine whether your model might be affected by these bugs.

Input Parameters

To display bug reports modified after a certain date, use the **Only show bug reports modified after date (mm/dd/yyyy)** field.

Results and Recommended Actions

Condition	Recommended Action
There are bug reports for the Simulink Verification and Validation R2013a product.	Review the bug report descriptions and workarounds provided in the links listed in the **ID** column of the Model Advisor window.

See Also

"Simulink Verification and Validation" documentation

Display bug reports for Simulink Design Verifier

Display bug reports for the Simulink Design Verifier R2013a product.

Description

Run this check to display the bug reports for Simulink Design Verifier R2013a that are available at www.mathworks.com/support/bugreports.

Note This check does not determine whether your model might be affected by these bugs.

Input Parameters

To display bug reports modified after a certain date, use the **Only show bug reports modified after date (mm/dd/yyyy)** field.

Results and Recommended Actions

Condition	Recommended Action
There are bug reports for the Simulink Design Verifier R2013a product.	Review the bug report descriptions and workarounds provided in the links listed in the **ID** column of the Model Advisor window.

See Also

"Simulink Design Verifier" documentation

Display bug reports for Simulink PLC Coder

Display bug reports for the Simulink PLC Coder R2013a product.

Description

Run this check to display the bug reports for Simulink PLC Coder R2013a that are available at www.mathworks.com/support/bugreports.

Note This check does not determine whether your model might be affected by these bugs.

Input Parameters

To display bug reports modified after a certain date, use the **Only show bug reports modified after date (mm/dd/yyyy)** field.

Results and Recommended Actions

Condition	Recommended Action
There are bug reports for the Simulink PLC Coder R2013a product.	Review the bug report descriptions and workarounds provided in the links listed in the **ID** column of the Model Advisor window.

See Also

"Simulink PLC Coder" documentation

Display bug reports for IEC Certification Kit

Display bug reports for the IEC Certification Kit R2013a product.

Description

Run this check to display the bug reports for IEC Certification Kit R2013a that are available at www.mathworks.com/support/bugreports.

Note This check does not determine whether your model might be affected by these bugs.

Input Parameters

To display bug reports modified after a certain date, use the **Only show bug reports modified after date (mm/dd/yyyy)** field.

Results and Recommended Actions

Condition	Recommended Action
There are bug reports for the IEC Certification Kit R2013a product.	Review the bug report descriptions and workarounds provided in the links listed in the **ID** column of the Model Advisor window.

See Also

"IEC Certification Kit (for ISO 26262 and IEC 61508)" documentation

Display bug reports for Polyspace products for C/C++

Display bug reports for the Polyspace Client for C/C++ and Polyspace Server for C/C++ R2013a products.

Description

Run this check to display the bug reports for Polyspace Client for C/C++ and Polyspace Server for C/C++ R2013a that are available at www.mathworks.com/support/bugreports.

Note This check does not determine whether your model might be affected by these bugs.

Input Parameters

To display bug reports modified after a certain date, use the **Only show bug reports modified after date (mm/dd/yyyy)** field.

Results and Recommended Actions

Condition	Recommended Action
There are bug reports for the Polyspace Client for C/C++ R2013a product.	Review the bug report descriptions and workarounds provided in the links listed in the **ID** column of the Model Advisor window.
There are bug reports for the Polyspace Server for C/C++ R2013a product.	Review the bug report descriptions and workarounds provided in the links listed in the **ID** column of the Model Advisor window.

See Also

“Polyspace Products for C/C++ Documentation”

Display bug reports for Embedded Coder

Display bug reports for the Embedded Coder R2013a product.

Description

Run this check to display the bug reports for Embedded Coder R2013a that are available at www.mathworks.com/support/bugreports.

Note This check does not determine whether your model might be affected by these bugs.

Input Parameters

To display bug reports modified after a certain date, use the **Only show bug reports modified after date (mm/dd/yyyy)** field.

Results and Recommended Actions

Condition	Recommended Action
There are bug reports for the Embedded Coder R2013a product.	Review the bug report descriptions and workarounds provided in the links listed in the **ID** column of the Model Advisor window.

See Also

"Embedded Coder" documentation