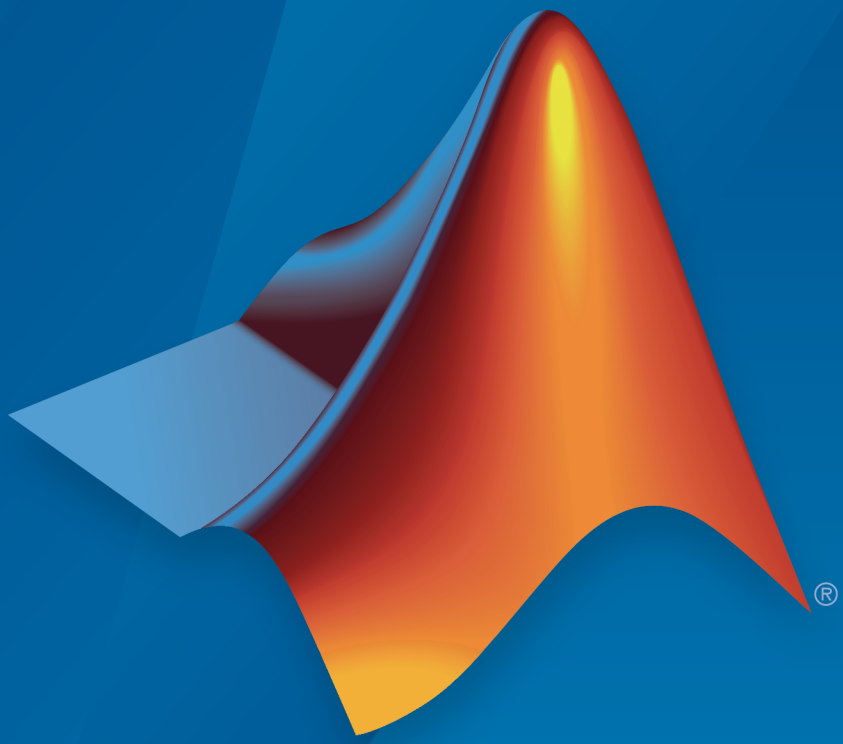
Simulink® Test™ Release Notes



MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Simulink® Test™ Release Notes

Contents

R2017a

R2016b

R2015aSP1

R2015b

R2015a

R2017a

Harness Import: Simplify test harness management by converting test harness models to Simulink Test harnesses

To simplify management and synchronization of standalone harness models, such as those created manually, using Simulink® Verification and Validation™, or Simulink Design Verifier™, convert the standalone models to test harnesses in Simulink Test™. Converting standalone models to test harnesses allows you to iterate on your design using push and rebuild functions, and manage test harnesses using the UI and programmatic interface.

For more information, see "Create Test Harnesses from Standalone Models" and the function `sltest.harness.import`.

Expanded Test Harness Capabilities: Create test harnesses with additional semantics, blocks, sources, and synchronization behavior

1. Control when the component under test design synchronizes between the model and harness. When creating a test harness, select the harness **Synchronization Mode** in the **Advanced Properties**. For an existing harness, click the harness properties badge to change the synchronization mode. For more information, see "Synchronize Changes Between Test Harness and Model". Synchronization modes include:

   - `Synchronize on harness open and close`: The component in the main model, or the test harness, is automatically updated when the harness closes or opens.
   - `Synchronize on harness open`: The component in the test harness is updated when the harness opens.
   - `Synchronize only during push and rebuild`: Synchronization does not occur when the harness opens or closes. You manually control synchronization by selecting **Analysis > Test Harness > Push Component and Parameters to Main Model** or **Analysis > Test Harness > Rebuild Harness from Main Model**.

2. Use Signal Builder blocks as sources for test inputs generated by Simulink Design Verifier analysis. `Signal Builder` is a **Harness Source** option when you select **Export test cases to Simulink Test** in the Simulink Design Verifier results dialog box. For more information, see "Test Models Using Inputs Generated by Simulink Design Verifier".

3. Test user-defined functions by creating test harnesses for the following additional blocks in the User-Defined Functions library:
   - S-Function block
   - S-Function Builder block
   - Level-2 MATLAB S-Function block
4. Create test harnesses for subsystems within linked library blocks.
5. Move and clone test harnesses across different blocks in the same model or in different models. See "Move and Clone Test Harnesses" and the functions `sltest.harness.move` and `sltest.harness.clone`.
6. Create test harnesses for models that use a mix of asynchronous and rate-based modeling.
7. Create test harnesses that honor the sorted execution order of blocks and subsystems for export function modeling.

Compatibility Considerations

For `sltest.harness.create` and `sltest.harness.set`, the `'EnableComponentEditing'` option is removed. Editing the component under test is controlled by the CUT synchronization mode. Update scripts that use the `'EnableComponentEditing'` option to specify `'SynchronizationMode'`.

- To prevent editing the component under test in the test harness (previously specified by `'EnableComponentEditing',false`), use `'SynchronizationMode','SyncOnOpen'`.
- To allow editing the component under test in the test harness (previously specified by `'EnableComponentEditing',true`), use `'SynchronizationMode','SyncOnOpenAndClose'` (default) or `'SynchronizationMode','SyncOnPushRebuildOnly'`.

Support for Debug Mode: Debug system under test simulation

From the Test Manager, you can debug the system under test simulation. Click the **Debug** button in the Test Manager toolstrip before executing the test. The simulation pauses at simulation start and presents a debug prompt at the command line. You can step through simulation using the Stepping Options buttons in the Simulink Editor toolstrip. For more information, see "Use Simulation Stepper" (Simulink).

Time Tolerance: Specify leading and lagging time tolerance in test cases

Account for temporal shifts in your test results using time tolerances. Some test cases, such as real-time tests, are affected by timing attributes of the execution environment and shifts in data logged from physical systems. You can account for timing differences by including time tolerance in baseline and equivalence criteria. For details, see "Apply Tolerances to Test Criteria".

Simscape Component Support: Create test harnesses for subsystems connected by Simscape physical connections

You can test subsystems connected by Simscape™ physical connections using test harnesses. Test harnesses isolate Simscape subsystems, providing physical signal ports at the subsystem interface. To define the physical system, add blocks to the test harness. To connect Test Sequence and Test Assessment blocks to the physical system, use Simulink-PS Converter and PS-Simulink Converter blocks.

Expanded API: Edit and manage Test Sequence Blocks with an expanded API

The `sltest.testsequence` API includes additional functions to read, edit, and delete test sequence steps, transitions, and data symbols. Use these functions to create, edit, and manage Test Sequence and Test Assessment blocks. For more information, see the Test Sequence Programming section on the "Logic-Based Testing" page.

The `'Label'` property name has been changed to `'Action'` to more closely match the functionality of the argument in creating and editing test sequence steps. The change applies to the functions

- `sltest.testsequence.addstepafter`
- `sltest.testsequence.addstepbefore`
- `sltest.testsequence.addstep`
- `sltest.testsequence.editstep`

Compatibility Considerations

If you have a script that uses these functions with the property name `'Label'`, running the script returns a warning that `'Label'` is removed. Update the script to use the property name `'Action'` instead of `'Label'`.

Signal Logging Selection: Specify signals to log in a test case without modifying the model

Using the **Simulation Outputs** section, you can log additional signals in the Test Manager without changing the logging settings in your model. For more information, see "Simulation Outputs".

Proof Objective Support for `verify()` Statements: `verify()` statements interpreted as proof objectives for Simulink Design Verifier analysis

If your model or test harness contains a `verify()` statement in a Test Assessment or Test Sequence block, Simulink Design Verifier property proving analysis interprets the `verify()` statement as a proof objective. This allows your `verify()` statements to be used for both functional testing and formal analysis, without having to add Proof Objective blocks to the model. Also, for `verify()` statements falsified, you can create counterexamples that falsify the objective during simulation. For more information about property proving, see "Prove Properties in a Model" (Simulink Design Verifier). For more information about `verify()` statements, see "Assess Simulation Using Logical Statements".

Enhanced Reporting: Include MATLAB figures, save report options, and automatically generate reports after testing

- You can include custom MATLAB® figures in your report. For details, see "Create, Store, and Open MATLAB Figures".
- You can automatically create a report after executing a test file. In the test file, under **Test File Options**, select **Generate report after execution**. The Test Manager displays options for the report, which are saved with the test file. For details, see "Export Test Results and Generate Reports".

Test Manager Integration with Simulink Design Verifier: Generate additional tests from within the Test Manager to increase coverage

If you have Simulink Design Verifier, you can generate tests to achieve additional coverage starting from the coverage results pane in the Test Manager. After executing

tests, view the cumulative coverage results in the Test Manager results pane. Select the coverage result and click **Add Tests for Missing Coverage**. For an example, see "Perform Functional Testing and Analyze Test Coverage".

Test Manager Hierarchies: Expand and collapse trees using context menu

You can expand or collapse hierarchies in the Test Manager using the context menu. For example, in the **Test Browser** pane, right-click a test file or test suite. From the context menu, select **Expand All** or **Collapse All**. A similar context menu item appears in hierarchies in the **Results and Artifacts** pane.

Stop simulation at the end of test input data

If you have timeseries test input data, you can limit the simulation data output by stopping simulation at the end of the test input timeseries. For example, if your input data stops after 10 seconds, but your model simulation time is set to 300 seconds, limit the simulation to avoid 290 seconds of unnecessary data. Select **Stop simulation at last time point** in the **Inputs** section of the test case definition.

Capture baseline data from test case iterations

If your test case is configured with table iterations, you can capture baseline data from logged signals using a one-click process. Baseline criteria is captured in a separate file for each iteration, and each baseline data file appears with its corresponding iteration in the iterations table. For an example, see "Capture Baseline Data from Iterations".

Simplify test editor view by hiding unused sections

You can simplify the view for test files, test suites, and test cases by hiding unused test sections. In the Test Manager, click the **Preferences** button in the toolbar, and select sections to hide or display. Populated sections are always displayed. This is a global Test Manager setting. To customize view for multiple users, you can set the preferences programmatically using `sltest.testmanager.getpref` and `sltest.testmanager.setpref`. For more information, see "Test Sections".

Test Case Conversion: Convert desktop simulation test cases to real-time test cases

You can convert test cases that run on desktop simulation into real-time test cases. In the **Test Browser**, right-click the test case name and select **Convert to > Real-Time Test**.

Data dimension settings for certain test harness inputs

When you create a test harness using From Workspace, From File, or Constant blocks as sources, the default value of the source reflects dimensions of the signal.

Run MATLAB Unit Tests in Fast Restart mode

The MATLAB Unit Test framework supports Fast Restart mode for running test cases authored in Simulink Test.

R2016b

Version: 2.1

New Features

Bug Fixes

Custom Criteria: Define custom criteria for test case evaluation

You can customize test case pass and fail criteria using MATLAB and the MATLAB Unit Testing framework. Use custom criteria in addition to assessments such as `verify` statements and timeseries comparisons. For example, assess the final value of a signal, set a maximum threshold, or post-process signal results using MATLAB toolboxes. With MATLAB Unit Test, qualifications return `pass` or `fail` results to the Test Manager. See Apply Custom Criteria to Test Cases.

MATLAB Unit Test Integration: Use MATLAB Unit Test to execute and integrate tests

You can run tests authored in Simulink Test using the MATLAB Unit Test framework. This allows you to combine execution of tests authored in both frameworks. You can customize test execution with a test runner, and access test results in MATLAB.

With the MATLAB Unit Test framework, you can set up systematic testing using continuous integration systems. Use plugins such as the `TAPPPlugin` to create results that are compatible with CI systems such as Jenkins™. See Test Models Using MATLAB Unit Test.

Test Case Tagging: Filter and execute tests using custom tags

You can group tests by assigning custom tags to test cases and suites, and filter tests to view test case subsets. Running filtered tests can save time compared to running a full test suite. Filter tests in the test browser using the syntax `tag:<name>`. To run filtered tests, expand the drop-down menu under **Run** and select **Run filtered**. See Filter Test Execution and Results.

Graphical Output for Model Verification Blocks: Analyze results from Assertion and Model Verification blocks in Simulation Data Inspector and the Test Manager

Blocks in the Model Verification library return a `pass` or `fail` result to the Test Manager, using semantics similar to a `verify` statement. Viewing results graphically helps you to:

- Determine the time when a failure occurs.

- Debug the model by comparing the verification result with relevant signals.
- Trace failures from the graphical results to the model.

See View Graphical Results From Model Verification Library.

Initialize and Terminate Functions: Test harnesses include calls for initialize and terminate systems in a model

When you create a test harness for a model block diagram, you can include calls to initialize and terminate systems. The test harness creates a Test Sequence block configured to schedule function calls to initialize and terminate systems.

Harness Requirements Linking: Establish requirements traceability for external test harnesses

Requirements linking is supported for test harnesses that are stored externally as independent SLX files.

Test Case Conversion: Convert test cases between baseline, equivalence, and simulation test types

You can convert existing test cases between baseline, equivalence, and simulation test types. This helps facilitate test case reuse. For example, to reuse an existing baseline test as an equivalence test, copy the baseline test and change the copied test case to an equivalence test. To convert a test case,

1. In the Test Browser, right-click the test case name.
2. Select **Convert to > Baseline Test / Equivalence Test / Simulation Test**.

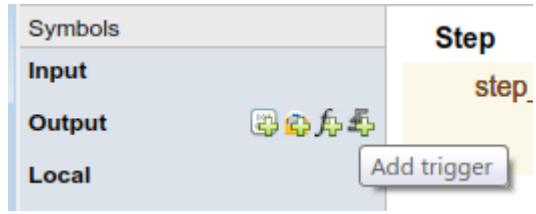
Additional Report Templates: Create PDF and HTML reports using report templates

With a MATLAB Report Generator™ license, you can create custom PDF and HTML reports from the Test Manager using report templates. In the Create Test Result Report dialog box, select a PDFTX or HTMTX template file. For more information, see Export Test Results and Generate Reports, and Create Report Templates.

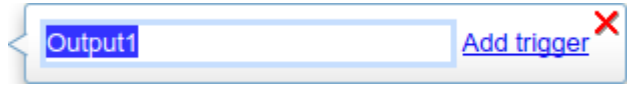
Output Event Definition: Create Test Sequences that trigger actions in other subsystems

You can use trigger outputs in a Test Sequence block or Test Assessment block to activate a triggered subsystem or signal an event in a Stateflow® chart. To create a trigger output in a Test Sequence block,

1 In the test sequence editor **Symbols** pane, click the **Add trigger** icon next to the **Output** section.



2 Enter the output name, and click **Add trigger**.



3 Trigger outputs initialize to 0. In the test sequence, use the `send` command to activate the trigger output.

```
send(Output1)
```

Access `verify` statement output using a programmatic interface

Simulation data output from `verify` statements are available via a programmatic interface. To get assessment results, use the sltest.getAssessments function.

Revised visualization for `verify` statements and other assessments

The Test Manager displays `verify` statement results and other assessments using a stem plot, which plots the `verify` output data as stems extending from a baseline along the x-axis. The baseline is a `pass` result, with stems extending from the baseline for `untested` and `fail` results. Data color corresponds to the statement result:

Color	Result
Red	`Fail`
Green	`Pass`
Gray	`Untested`

To maintain readability, plots adjust the data displayed if the result is the same for numerous consecutive data points. Zooming in to smaller x-axis intervals displays additional discrete data points. Changes in result are always displayed, for example, from `pass` to `fail`. See Assess Simulation Using Logical Statements for an example. To get assessment results programmatically, use the sltest.getAssessments function.

Common test harness sources and sinks for signal and control inputs, and revised signal routing

When you create a test harness, additional inputs are connected to the specified harness source and sink type, rather than to Inport or Outport blocks. Additional inputs expand your ability to drive component inputs with a single source type for:

- Control inputs
- Function call inputs
- Data store memory
- Goto and From blocks
- Export function models
- Simulink functions

To simplify test harness block diagrams, the signal routing uses Goto and From blocks to connect component under test input and output signals to Test Assessment blocks. Also, some signal routing blocks are contained in input and output conversion subsystems.

Add baseline criteria using expected outputs captured by Simulink Design Verifier

Simulation output captured by running Simulink Design Verifier tests are now included as baseline data in test cases exported from Simulink Design Verifier to Simulink Test.

Set SIL or PIL simulation mode from the Test Manager for a model referenced by a test harness

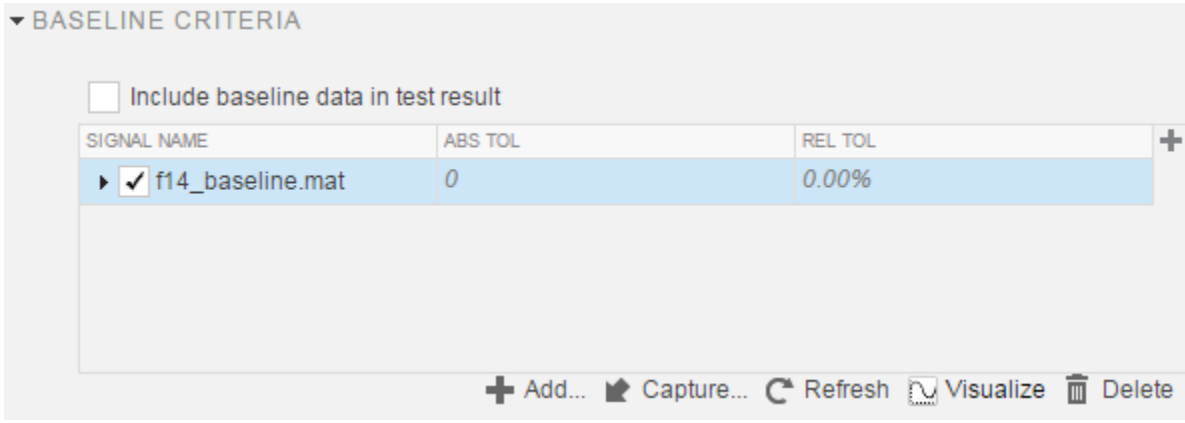
From the Test Manager, you can override the simulation mode to software-in-the-loop (SIL) or processor-in-the-loop (PIL) for a model referenced by a component under test in a test harness. Overriding the referenced model simulation model applies to test harnesses for block diagrams and test harnesses for Model blocks, since these types of test harnesses use Model blocks as the component under test. Setting the simulation mode from the Test Manager allows you to use an equivalence test and a single test harness to perform SIL or PIL output equivalence verification.

Highlight dependencies in test harnesses using Model Slicer

If you have a Simulink Design Verifier license, you can use the Model Slicer to highlight functional dependencies in test harnesses created by Simulink Test. Highlighting functional dependencies helps you analyze behavior in large or complex test harnesses. See Highlight Functional Dependencies.

View baseline data graphically

In the Test Manager, you can view a graph of test case baseline data. This facilitates reviewing the baseline data before you map it to a test case and run tests. In the **Baseline Criteria** section of the test case, highlight the signal name and click **Visualize**. The graph appears in the Simulation Data Inspector.



R2016a

Version: 2.0

New Features

Bug Fixes

Real-Time Testing: Author and execute real-time tests with Simulink Real-Time

A new Real-Time Test builds a Simulink Real-Time™ application from your model or test harness and runs it on a target computer. You can assess the real-time execution using `verify` statements, and collect real-time data for analysis in the Test Manager. See Test Models in Real Time.

`verify` Statement: Author test sequence assessments to verify simulation behavior without stopping the simulation

In the Test Sequence and Test Assessment blocks, you can use `verify` statements to assess a logical condition without stopping simulation. A `verify` statement returns a fail, pass, or untested result. Results of each `verify` statement appear in the Test Manager. See Assess Simulation Using Logical Statements.

Test Report Customization: Customize test result reports using Simulink Report Generator

You can write scripts to customize the details of Test Manager result reports such as text formatting, output plots, headers and footers, layouts, and more. See Customize Generated Reports.

External Test Harnesses: Save Simulink Test harnesses as external files

You can opt to save your test harnesses externally, as independent SLX files. External test harnesses allow you to create or change test harnesses without changing the model SLX file, which is useful for models under change management. External harnesses provide the same synchronization and push/rebuild capability as internal harnesses saved with the model SLX file. See Manage Test Harnesses.

Test Iterations: Create tests with iterations such as parameters and input vectors

To test and sweep through a range of parameters, inputs, and other test case settings, you can author and organize many tests in one place using iterations. To help create

iterations, templates are available for Signal Builder groups, parameter sets, inputs, configuration settings, and baseline criteria. You can run iterations using fast restart if it is supported by your model. See Run Multiple Combinations of Tests Using Iterations.

Test generated using Simulink Design Verifier now appear as iterations in a test case rather than separate test cases in a test suite.

Aggregate Coverage Results: Aggregate Simulink Verification and Validation coverage results across executed tests

If you have a Simulink Verification and Validation license, then you can collect the coverage results from your tests. The results are aggregated at test case, test suite, and test file levels. Coverage results can also be included in the Test Manager results report. See Collect Coverage in Tests.

Parallel Test Execution: Distribute test execution in parallel to decrease test run time

If you have a Parallel Computing Toolbox™ license, then you can run tests in parallel across multiple workers to decrease test execution time. See Run Tests Using Parallel Execution.

Test Harness for Libraries: Create and manage test harnesses for library components

You can create test harnesses for library blocks and move test harnesses from linked blocks to the library source. See Test Library Blocks.

Requirement Traceability: Link to requirements in test harnesses and test sequences

If you have a Simulink Verification and Validation license, you can create requirements links for model objects in internally stored test harnesses. Requirement links for the component under test synchronize between the main model and the test harness. You can also create requirements links for test steps in Test Sequence and Test Assessment blocks. See Link Tests to Requirements.

Simulink and Export Function Support: Create test harnesses for models containing Simulink functions and export functions

If you generate a harness for a model configured to export functions, the harness will contain a new Test Sequence block that schedules the function-call signals and Simulink Functions in the export-function model. You choose the sources and sinks for other subsystem inputs and outputs. See Test Models that use Export Functions for AUTOSAR-Compliant Code.

Test Assessment block available for all harness sources

The Simulink Test library offers a separate Test Assessment block entry. You can include a Test Assessment block with any test harness source using the test harness creation dialog box. The Test Assessment block is a Test Sequence block configured with a default When decomposition sequence and a `verify` statement, which are commonly used in model assessment.

Test Sequence block support for messages

Test Sequence blocks support sending and receiving messages. Messages are objects that carry data and can be queued. You can send a message using a message output and the `send` command, and receive a message using a message input and the `receive` command. When a test step receives a message, it can use the `receive` result or the message data in a step action or transition. See Test Sequence Action and Transition Operations.

Test Sequence Editor enhancements

The Test Sequence Editor offers several enhancements for the Test Sequence and Test Assessment blocks.

- You can add a description for a test step using the **Description** field.
  - Code generated from the block includes test step descriptions as commented code. To include the commented descriptions in generated code, select **Simulink block descriptions** in the **Code Generation** > **Comments** section of the model configuration parameters.
  - Simulink Report Generator includes descriptions in the Test Sequence block reports.

- Syntax highlighting: The Test Sequence Editor includes MATLAB syntax highlighting for improved readability.
- Tab completion: The Test Sequence Editor suggests words, such as data symbols and functions, to complete test step programming syntax. A list appears based on the characters you type. Select a word from the list and press **Tab**, or press **Esc** to close the suggestions.
- Port reordering: You can reorder block inputs and outputs by dragging an **Input** or **Output** symbol name up or down in the **Symbols** sidebar.

Simulink Projects integration

You can create a Simulink project from a test file. When you create a project from a test file, it enables you to perform file dependency analysis. Projects let you easily see the impact that changes could have on tests that might use shared files. You can also run tests directly from the Simulink Projects interface. For more information about Simulink Projects integration, see Manage Test File Dependencies.

Test Generation for Subsystems

You can generate tests for a subsystem in a model from the Test Manager. The Test Manager creates a test harness for the subsystem and enables you to test the subsystem independently, thereby isolating it from the main model. See Generate Test Cases from Model Components.

R2015aSP1

Version: 1.0.1

Bug Fixes

R2015b

Version: 1.1

New Features

Bug Fixes

Expanded Simulink Test API: Automate test creation, editing, and execution using MATLAB scripts

You can use functions, classes, and methods to programmatically:

- Generate test cases from a model based on existing test harnesses and signal builder groups. See sltest.testmanager.createTestsFromModel.
- Edit test case simulation properties, parameter sets, and comparison criteria
- Create test files, test suites, and test cases
- Copy and move test suites and test cases
- Run individual test suites and test cases
- Copy a test harness, including its contents, configuration set, properties, and model association, using the sltest.harness.clone function

For more information about using the API, see Automate Tests Programmatically.

Test Case Automation: Create test cases with inputs generated by Simulink Design Verifier

Starting with the results of a Simulink Design Verifier analysis, Simulink Test creates test cases that use the inputs generated by Simulink Design Verifier. Test cases appear in the Test Manager and can use an existing or new test harness. See Test Models Using Inputs Generated by Simulink Design Verifier.

Qualification and Certification: Qualify Simulink Test for supported industry standards, including DO-178 and ISO 26262

You can use the IEC Certification Kit and DO Qualification Kit to qualify Simulink Test for supported industry standards, including DO-178, ISO 26262, and IEC 61508.

Enhanced Reporting: Use Microsoft Word templates to customize report generation

If you have a MATLAB Report Generator license, you can insert report items from the Simulink Test generated report into your own Microsoft® Word templates. For more information on report generation, see Export Test Results and Generate Reports.

Additional tools for Test Sequence editing and debugging

The test sequence editor includes new tools you can use when creating, editing, and debugging a test sequence, including

- Undo and redo edits
- Cut, copy, and paste test steps using keyboard shortcuts
- Simulation rollback

Simulink Report Generator inclusion for Test Sequence block

You can include data from Test Sequence blocks in a report, using the Test Sequence component in Simulink Report Generator.

R2015a

Version: 1.0

New Features

Introduction to Simulink Test

Simulink Test provides tools for authoring, managing, and systematically executing simulation-based tests. You can create nonintrusive test harnesses to test models and subsystems. You can generate reports, archive and review test results, rerun failed tests, and debug the component or system under test.

Test harness for subsystem and model testing

Test harnesses provide a separate, nonintrusive testing environment for your models. A test harness associates with a particular model or model component and persists with the model. You define tests by adding inputs and assessments to the harness, and you can set harness-specific simulation parameters. The test harness synchronizes model changes to the main model. The Test Manager can access the test harnesses in your model. See Refine, Test, and Debug a Subsystem and Test Harness and Model Relationship.

Test Sequence block for defining tests and assessments

Test Sequence blocks concisely define a series of test steps and transitions using MATLAB action language. Each step defines the block output values and the condition that triggers the transition to another test step. You can define a test step hierarchy using different transition modes. Test Sequence blocks include concise output functions, such as square and sawtooth, and operators that return temporal information, such as the elapsed step time or the duration of a condition.

You can assess the model operation in the test sequence, or in a separate Test Sequence block. See Test Downshift Points of a Transmission Controller and the Test Sequence block.

Test Manager for test authoring and systematic test execution

The Simulink Test Manager enables you to organize and run large sets of tests for Simulink models. Using the Test Manager, you can author and execute test cases individually or as a batch. You can also link to test requirements from each test case if you have a Simulink Verification and Validation license. After you execute tests, the test outcome and any simulation output appear in the **Results and Artifacts** pane of the Test Manager.

Baseline, equivalence, and back-to-back testing with pass-fail criteria

You can test models using baseline and equivalence test case templates in the Test Manager. Baseline test cases compare simulation output to defined expected outputs. Equivalence test cases compare simulation output a second simulation. The simulation output comparison is evaluated according to absolute or relative tolerances, which you specify under **Baseline Criteria** or **Equivalence Criteria**. For more information on tolerances, see How Tolerances Are Applied to Test Criteria.

Archiving and reporting test cases and test results

After you execute tests, you can export the results in the **Results and Artifacts** pane of the Test Manager to a file or save them in a report. For more information on exporting results and generating reports, see Export Test Results and Generate Reports.