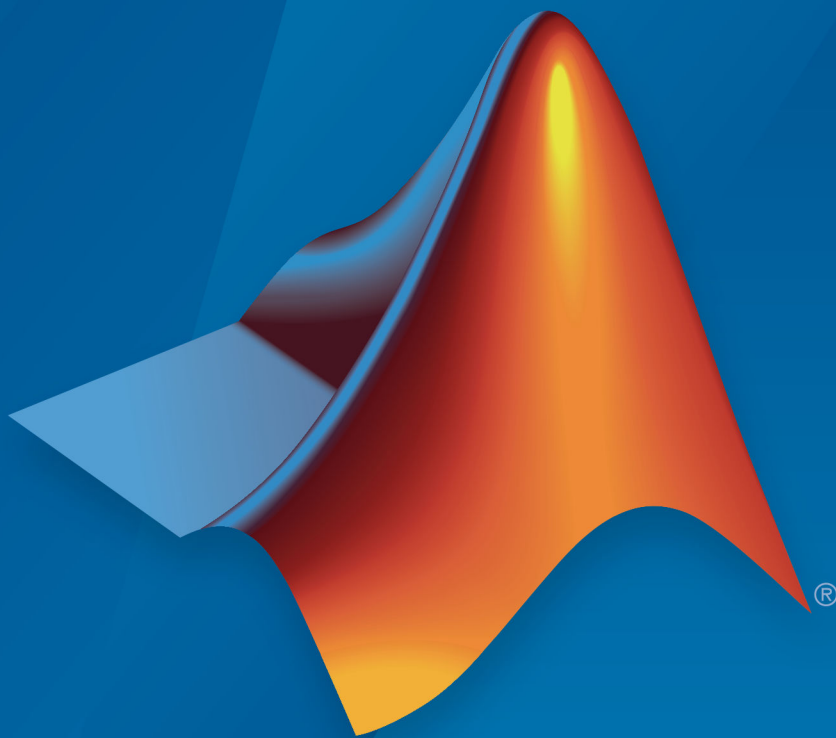


Simulink® Check™ Release Notes



MATLAB® & SIMULINK®

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.
1 Apple Hill Drive
Natick, MA 01760-2098

Simulink® Check™ Release Notes

© COPYRIGHT 2017-2019 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

R2019a

Model Slicer available with Simulink Check	1-2
Hierarchical view of metrics dashboard results	1-2
Filters for metrics dashboard results	1-2
Streamline compliance with modeling guidelines by using enhanced edit-time check diagnostics	1-3
Edit-Time check diagnostics interface	1-3
Model Advisor checks evaluated by using edit-time checking	1-3
JMAAB 5.1 Support: Automate checking of models to comply with JMAAB 5.1 modeling style guidelines	1-4
Additional MAAB 3.0 Checks: Improve quality and compliance to guidelines	1-6
Added functions for edit-time checking	1-7
Model optimization by sharing prelookup operation of n-D lookup tables	1-7
Clone metrics include exact and similar clones	1-7
Model slicer support for multi-instance model reference ...	1-8
Model Advisor option to compile model for code generation	1-8
Updates for verifying compliance with High-Integrity Systems Modeling guidelines	1-9

MISRA C:2012 and Secure Coding checks to improve compliance of generated code	1-12
Tech Preview of model refactoring using clone detection ...	1-12

R2018b

Metrics Dashboard Customization: Configure compliance metrics, add metric thresholds, and customize Metrics Dashboard layout	2-2
Configure Compliance Metrics	2-2
Add Metric Thresholds	2-2
Customize Metrics Dashboard Layout	2-3
Simscape Support with Clone Detection: Detect and refactor clones in Simscape Models	2-5
JMAAB 4.01 Support: Automate checking of models to comply with JMAAB 4.01 modeling style guidelines	2-5
Additional MAAB 3.0 and High Integrity Checks: Improve quality and compliance to guidelines	2-8
High Integrity Systems Modeling Checks: Use the additional conditions to check the configuration parameters	2-8
MISRA C:2012 and Secure Coding Standards: Improve compliance of generated code by using updated Model Advisor checks	2-9
Mnemonic Support: Use keyboard shortcuts with Metrics Dashboard	2-10
Check Style for Model Advisor: Create checks that generate interactive reports	2-10
Functionality Being Removed or Changed	2-15

Additional Checks for MAAB 3.0 and JMAAB 4.0 Guidelines: Automate checking for MAAB 3.0 guidelines for Simulink, Stateflow, Variant Subsystems, and MATLAB Function Blocks and JMAAB 4.0 guidelines	3-2
MAAB Modeling Checks	3-2
JMAAB Modeling Checks	3-5
Block Constraint Authoring with Edit-Time: Define checks for supported or unsupported blocks and parameters while editing	3-8
Clone Refactoring Workflow: Apply multiple refactoring steps to the same model	3-9
Automatic Refactoring for Similar Clones: Add masks to similar clones and refactor model	3-9
Clone Detection Exclusion Editor: Exclude subsystems and referenced models from clone detection	3-10
Automatic Data Store Memory Block Elimination: Identify and refactor Data Store Memory Block blocks with Model Transformer	3-10
Grid Visualization for Metrics: View results of Model Advisor checks in a grid to identify patterns in results	3-11
MathWorks High-Integrity Guidelines and Checks: Verify compliance with safety standards by using high-integrity checks and guidelines	3-12
High-Integrity System Modeling Checks	3-12
High-Integrity Modeling Guidelines	3-14
MISRA C: 2012 Modeling Checks: Improve compliance of generated code by using MISRA C:2012 standards checks	3-17
Secure Coding Modeling Checks: Update to Secure Coding compliance checks	3-18

Enhanced Edit-Time Checking Support: Edit-time checking for blocks not recommended for C/C++ production code deployment	3-19
Model Advisor Support for Inactive Variants: Run Model Advisor checks on active and inactive variants and generate report	3-19
Metric Engine Improvement: Collect and analyze metric data faster	3-19

R2017b

Simulink Verification and Validation Packaging: Moved compliance checking, model metrics, clone detection and refactoring, edit-time checking and model transformer to Simulink Check	4-2
Metrics Dashboard: Collect and view metric data for quality assessment	4-2
MathWorks High-Integrity Guidelines and Checks: Verify compliance with safety standards by using high-integrity checks and guidelines	4-3
Categorization of the Model Advisor Checks for High-Integrity Systems	4-3
High-Integrity Model Advisor Checks for DO-178C/DO-331 Standards	4-3
High-Integrity Model Advisor Checks for EN 50128, IEC 61508, IEC 62304, and ISO 26262 Standards	4-5
High-Integrity Modeling Guidelines	4-7
Modeling Support for Secure Coding Standards: Check model for compliance with secure coding requirements in CERT C, CWE, ISO/IEC TS 17961 standards to improve security of generated code	4-16

MISRA C: 2012 Modeling Checks: Improve compliance of generated code by using new MISRA C:2012 standards checks	4-19
DO-178C/DO-331 Modeling Checks: Removed Model Advisor check "Check model for block upgrade issues"	4-20
Model Metrics: Evaluate model quality by using new metric algorithms	4-21
Model Metric APIs: Create custom metrics with more detailed results and determine passed or failed compliance checks	4-21
Model Advisor Configuration Editor: Select edit-time checks from folders	4-22
Model Metric APIs: Removed Model block architectural component	4-23

R2019a

Version: 4.3

New Features

Bug Fixes

Model Slicer available with Simulink Check

Previously, the Model Slicer was available with Simulink® Design Verifier™. In R2019a, the tool is available with Simulink Check™. Use the Model Slicer to determine the interdependencies of blocks, signals, and model components throughout a model. Also, use Model Slicer to create simplified standalone models that are easier to understand and analyze yet retain their original context. To access the Model Slicer, select **Analysis > Model Slicer**. For more information, see “Model Simplification with Dependency Analysis”.

Hierarchical view of metrics dashboard results

In R2019a, you can view metric data for related components together by viewing Metric Dashboard results in a model hierarchical view. You can then better understand aggregated model metric data. To use the model hierarchical view, run the Metrics Dashboard on your model. Click a widget and then click the new **Tree** button. The figure shows metric results for the model `sf_car` in the Tree view.

Cyclomatic complexity			
Metric that calculates the cyclomatic complexity for model, subsystems and charts.			
COMPONENT	TYPE	MODEL COMPLEXITY	MODEL COMPLEXITY (INCL. DESCENDANTS)
▼ <code>sf_car</code>	Model	✔ 1	32
▼ <code>transmission</code>	SubSystem	✔ 0	5
Torque Converter	SubSystem	✔ 0	0
▼ <code>transmission ratio</code>	SubSystem	✔ 0	5
Look-Up Table	MATLABFunction	✔ 5	5
Vehicle	SubSystem	✔ 0	0
▼ <code>shift_logic</code>	Chart	✔ 20	26
▼ <code>selection_state.calc_th</code>	SubSystem	✔ 1	6
Look-Up	MATLABFunction	✔ 5	5
Engine	SubSystem	✔ 0	0

For more information on the Metrics Dashboard, see “Collect and Explore Metric Data by Using the Metrics Dashboard”.

Filters for metrics dashboard results

In R2019a, you can filter Metrics Dashboard results by component type, name, and path. After you run the Metrics Dashboard, to drill into a widget's metric details, select that

widget. In the **Table** view, select the context menu on the right side of the **TYPE**, **COMPONENT**, and **PATH** column headers. From the **TYPE** menu, select applicable components. From the **COMPONENT** and **PATH** menus, type a component name or path in the search bar. The Metrics Dashboard saves the filters for a widget, so you can view metric details for other widgets and return the filtered results. You cannot specify filters on the **High Integrity** and **MAAB** compliance widgets.

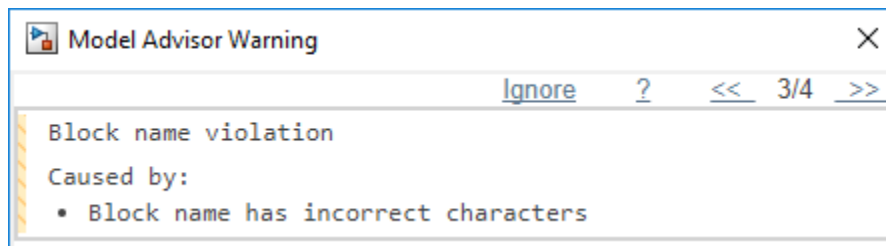
Filters enable you to quickly locate results. For example, to locate results for only Stateflow Charts, you can select that component from the **TYPE** menu. To analyze results for one Stateflow Chart, type its name or path in the **COMPONENT** or **PATH** search bar. For more information, see “Collect Model Metric Data by Using the Metrics Dashboard”.

Streamline compliance with modeling guidelines by using enhanced edit-time check diagnostics

Edit-Time check diagnostics interface

Enhancements to the edit-time checking capability make it easier for you to identify and resolve compatibility issues earlier in the model design process.

In R2019a, when you click the warning icon that appears over the highlighted block, the Model Advisor identifies compliance issues in the block that violate edit-time checks. When a block has multiple check violations, you can move between the violations by using the << and >> buttons. The cause of each issue is provided.



For information on checking your model while you edit, see “Check Model Compliance by Using the Model Advisor”.

Model Advisor checks evaluated by using edit-time checking

In R2019a, when you use edit-time checking, you can view violations of these Model Advisor checks. See the check documentation for additional information and limitations.

- “Check usage of non-compliant blocks”
- Check length of subsystem names
- Check length of Inport and Outport names
- Check length of block names
- Check operator order of Product blocks
- Check icon shape of Logical Operator blocks
- Check settings for data ports in Multiport Switch blocks
- Check usage of Switch blocks
- Check for unsupported block names
- Check for blocks not recommended for MISRA C:2012
- Check for switch case expressions without a default case
- Check usage of Assignment blocks
- Check for blocks not recommended for secure coding standards
- Identify lookup table blocks that generate expensive out-of-range checking code

JMAAB 5.1 Support: Automate checking of models to comply with JMAAB 5.1 modeling style guidelines

The JMAAB check group has been updated to support the latest release of the Japan MATLAB Automotive Advisory Board (JMAAB) guidelines 5.1. New checks have been added to support new guideline rules and removed when no longer applicable.

The following table identifies the new checks added to the JMAAB group for verifying compliance JMAAB 5.1. For more information, see Model Checks for Japan MATLAB Automotive Advisory Board (JMAAB) Guideline Compliance.

To execute these checks, open Model Advisor and select **By Task > Modeling Standards for JMAAB**.

Model Advisor Check	Check ID
Check default transition placement in Stateflow charts (JMAAB)	mathworks.jmaab.jc_0531
Check duplication of Simulink data names	mathworks.jmaab.jc_0791

Model Advisor Check	Check ID
Check for avoiding algebraic loops between subsystems	mathworks.jmaab.jc_0653
Check for unconnected objects in Stateflow Charts	mathworks.jmaab.jc_0797
Check for usage of text inside states	mathworks.jmaab.jc_0739
Check length of Stateflow data names	mathworks.jmaab.jc_0796
Check placement of Label String in Transitions	mathworks.jmaab.jc_0770
Check settings for data ports in Multiport Switch blocks	mathworks.jmaab.jc_0630
Check Stateflow chart action language	mathworks.jmaab.jc_0790
Check usable characters for Stateflow data names	mathworks.jmaab.jc_0795
Comparing floating point types in Simulink	mathworks.jmaab.jc_0800
Check signal line labels (JMAAB)	mathworks.jmaab.jc_0008
Check for propagated signal labels	mathworks.jmaab.jc_0009

The following table identifies checks that are no longer applicable to JMAAB 5.1 and have been removed from the JMAAB check group.

Check ID	Model Advisor Check
mathworks.jmaab.jc_0735	Check if each action in state label ends with a semicolon
mathworks.jmaab.jc_0737	Check if operators (ActionLanguage) for States and/or Transitions in Stateflow have uniform spaces around them.
mathworks.jmaab.jc_0742	Guidelines for writing boolean operations in condition labels in Stateflow transitions
mathworks.jmaab.jc_0743	Guidelines for writing condition actions in Stateflow transitions
mathworks.jmaab.jc_0756	Prohibited use of operation expressions in array indices

Check ID	Model Advisor Check
mathworks.maab.db_0151	Check transition actions in Stateflow charts
mathworks.maab.hd_0001	Check for prohibited sink blocks
mathworks.maab.himl_0003	Check MATLAB Function metrics
mathworks.maab.jc_0111	Check orientation of Subsystem blocks
mathworks.maab.jc_0221	Check character usage in signal labels
mathworks.maab.jm_0001	Check for prohibited blocks in discrete controllers
mathworks.maab.na_0008	Check signal line labels
mathworks.maab.na_0009	Check for propagated signal labels
mathworks.maab.na_0013	Check for comparison operations in Stateflow charts
mathworks.maab.jc_0281	Check Trigger and Enable block names

Additional MAAB 3.0 Checks: Improve quality and compliance to guidelines

This table identifies the new checks for verifying compliance with MATLAB Automotive Advisory Board Checks 3.0 (MAAB) guidelines. For information about MAAB guidelines, see Model Advisor Checks for MAAB Guidelines.

Model Advisor Check	Check ID
Check Simulink signal appearance	mathworks.maab.db_0032
Check for Stateflow transition appearance	mathworks.maab.db_0129
Check usage of Goto and From blocks between Subsystems	mathworks.maab.jc_0171
Check reuse of Variables within a Stateflow scope	mathworks.maab.jc_0491
Check usage of non-compliant blocks	mathworks.maab.na_0027
Check usage of enumerated values	mathworks.maab.na_0031

For more information, see “Model Checks for MathWorks Automotive Advisory Board (MAAB) Guideline Compliance”.

Added functions for edit-time checking

In R2019a, there are two new edit-time checking functions. Use the new `edittime.setAdvisorChecking` method to check for Model Advisor violations while you edit. The `edittime.setAdvisorChecking` method is equivalent to selecting **Analysis > Model Advisor > Display Advisor Checks in Editor**. Use the new `edittime.getAdvisorChecking` method to find out if Edit-time checking is on or not. For more information on edit-time checking, see “Check Model Compliance by Using the Model Advisor”.

Model optimization by sharing prelookup operation of n-D lookup tables

In R2019a, you can use the Model Transformer tool to improve the simulation speed for models with n-D Lookup table blocks by reusing the Prelookup operation with multiple blocks. This transformation results in decreased ROM usage and increased execution speed in the generated code.

You can run the **Transform table lookup into prelookup and interpolation** check on the Model Transformer tool to convert models.

A limitation is that the model transformation is confined to a single subsystem and does not extend beyond the subsystem boundaries. For further details, refer to “Improve Efficiency of Simulation by Optimizing Prelookup Operation of Lookup Table Blocks”

Clone metrics include exact and similar clones

Previously, when you ran the Metrics Dashboard on a model, the tool reported only exact clones in the **Potential Reuse** widget. Exact clones have identical block types, connections, and parameter values. When you clicked **Open Conversion Tool**, the Clone Detection tool ran and reported only exact clones because the **Maximum number of different parameters** parameter was set to 0 (default value). The Clone content and the Clone detection metrics included only exact clones as part of their calculations.

In R2019a, the Metrics Dashboard reports exact and similar clones in the **Potential Reuse** widget. Similar clones have identical block types and connections, but they can have different block parameter values. When you click **Open Conversion Tool**, the Clone Detection tool also reports exact and similar clones because the **Maximum number of different parameters** parameter is set to a high value. The Clone content and the Clone

detection metrics include exact and similar clones as part of their calculations. For more information on the Metrics Dashboard, see “Collect Model Metric Data by Using the Metrics Dashboard”.

Model slicer support for multi-instance model reference

You can now use model slicer on a Simulink model that contains multiple references to the same model in normal simulation mode.

Model Advisor option to compile model for code generation

In R2019a, the value 'PostCompileForCodegen' is added to the `ModelAdvisor.Check.CallbackContext` property. Use this option when developing custom Model Advisor checks to ensure code generation readiness of the model.

When using the existing 'PostCompile' option, the Model Advisor updates the model diagram and simulates the model. These Model Advisor checks do not flag modeling issues that fail during code generation because these issues do not affect the simulated model.

With the 'PostCompileForCodegen' option, the Model Advisor updates the model diagram specifically for code generation and does not assume that the model is being simulated. As a result, custom Model Advisor checks can identify code generation setup issues in a model at an earlier stage, avoiding unexpected errors during code generation.

The 'PostCompileForCodegen' option compiles for all variant choices in a model. This enables you to analyze possible issues present in the generated code for both active and inactive variant paths in the model. To compile for all variant choices, on the Variant system block, select the **Analyze all choices during update diagram and generate preprocessor conditionals** parameter.

Using the 'PostCompileForCodegen' option can increase the amount of time to execute your Model Advisor checks.

For additional information, see:

- `ModelAdvisor.Check.CallbackContext`
- “Define the Compile Option for Custom Checks”
- “Variant Systems” (Simulink)

Updates for verifying compliance with High-Integrity Systems Modeling guidelines

This table identifies the updates to the checks that verify compliance with High-Integrity Systems Modeling guidelines.

Model Advisor Check	Check ID	Description of Change
Check usage of relational operators in MATLAB Function blocks	mathworks.hism.himl_0008	Flags the MATLAB® relational operators that are used in functional format. For example:
Check usage of equality operators in MATLAB Function blocks	mathworks.hism.himl_0009	<ul style="list-style-type: none">lt - Less thanne - not equal to
Check usage of Bitwise Operator block	mathworks.hism.hisl_0019	Checks the Bitwise Operations for Signed data types .
Check usage of bitwise operations in Stateflow charts	mathworks.hism.hisf_0003	
Check Stateflow debugging options	mathworks.hism.hisf_0011	Checks the parameter Underspecified and Overspecified in the Settings menu of the Truth Table Editor , to Error .

Model Advisor Check	Check ID	Description of Change
Check usage of conditionally executed subsystems	mathworks.hism.hisl_0012	Extends support on the following sample-time dependent blocks: <ul style="list-style-type: none">• Discrete Filter• Discrete Fir• Discrete State Space• Discrete Transfer Fcn• Discrete Zero Pole• Discrete Integrator• First Order Transfer Fcn• Lead or Lag Compensator• Transfer Fcn Real Zero• Discrete Derivative• Discrete Transfer Function with Initial Outputs• Discrete Transfer Function with Initial States• Discrete Zero-Pole with Initial Outputs• Discrete Zero-Pole with Initial States
Check usage of Merge blocks	mathworks.hism.hisl_0015	Checks the Outport block parameter Output when disabled to held for each conditionally executed subsystem being merged.

Model Advisor Check	Check ID	Description of Change
Check for inconsistent vector indexing methods	mathworks.hism.hisl_0021	<p>Extends support on the following:</p> <p>Blocks that support configurable indexing:</p> <ul style="list-style-type: none"> • Index Vector • Multiport Switch • Assignment • Selector • For Iterator <p>Blocks that support only one-based indexing:</p> <ul style="list-style-type: none"> • Fcn • MATLAB Function • MATLAB System • State Transition Table • Test Sequence • Truth Table block • Stateflow chart with MATLAB action language • Truth Table function with MATLAB action language <p>Blocks that supports only zero-based indexing:</p> <ul style="list-style-type: none"> • Stateflow chart with C action language • Truth Table function with C action language
Check usage of shift operations for Stateflow data	mathworks.hism.hisf_0064	Does not check the negative values.

For more information, see “Model Checks for High Integrity Systems Modeling Checks”.

MISRA C:2012 and Secure Coding checks to improve compliance of generated code

Modifications to existing Model Advisor checks that you use to verify compliance with MISRA C:2012 and Secure Coding standards are outlined in this table.

Model Advisor Check	Description of Change
Check configuration parameters for MISRA C:2012	Checks now analyze the setting for these configuration parameters:
Check configuration parameters for secure coding standards	<ul style="list-style-type: none"> • Include comments (GenerateComments) • MATLAB user comments (MATLABFcnDesc)
Check for missing error ports for AUTOSAR receiver interfaces	<p>When an error port is missing, the check flags receiver interface ports with these AUTOSAR data access mode types:</p> <ul style="list-style-type: none"> • ImplicitReceive • ExplicitReceive • EndToEndRead

Tech Preview of model refactoring using clone detection

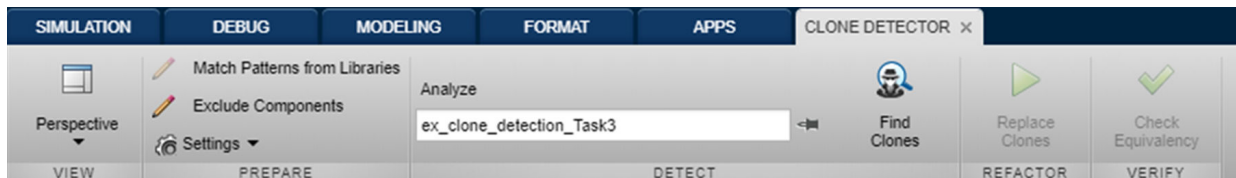
In R2019a, you can use the Clone Detector app that has a new and improved user interface. This user interface facilitates:

- Display of clones detected hierarchically as model view, tree view, list view, and clones plot.
- Displays the report from the clone refactorization and enables you to restore the model to its original state.

The Clone Detector app is a technical preview in R2019a. To try it out, turn on the Simulink Toolstrip. For more information, see “Simulink Toolstrip Tech Preview replaces menus and toolbars in the Simulink Desktop” (Simulink).

The documentation does not reflect the addition of the Clone Detection app. There is documentation for the existing Clone Detection tool. For more information, see “Enable Component Reuse by Using Clone Detection”.

The new user interface contains 5 distinct sections representing the workflow as shown below :



To access the parameter settings for help, results, and model properties of the Clone Detector app, click the **Perspective** menu.

You can use the settings present on the **PREPARE** tab to set the conditions for clone detection.

To identify clones in the model, click the **Find Clones** icon. The percentage of overall, exact, and similar clones that can be replaced by library blocks is displayed.

To refactor the models by creating links to library blocks of similar clones, click the **Replace clones** button. Backup models are created in case you want to restore models to their original configurations.

To open the Test Manager, that tests the equivalence between the original and modified model, click the **Check Equivalency** button.

R2018b

Version: 4.2

New Features

Bug Fixes

Compatibility Considerations

Metrics Dashboard Customization: Configure compliance metrics, add metric thresholds, and customize Metrics Dashboard layout

Configure Compliance Metrics

In R2018b, you can use the Metrics Dashboard and metric APIs to obtain compliance and issues data on your Model Advisor configuration. To set up your Model Advisor configuration, see [Organize Checks and Folders Using the Model Advisor Configuration Editor](#). After you have set up your configuration, create an `slmetric.config.Configuration` object. Use the `setMetricFamilyParameterValues` method to specify the check group for which you want to obtain compliance and issues data.

This code sample is for one check group.

```
CONF = slmetric.config.Configuration.new('Name', 'config.xml');
CONF.setMetricFamilyParameterValues('ModelAdvisorStandard',...
    {'SysRoot_$optimization_checks'})
CONF.save('FileName', 'config.xml')
slmetric.config.setActiveConfiguration(fullfile(pwd, 'config.xml'));
```

`SysRoot_$optimization_checks` is the Check Group ID. In the Model Advisor Configuration Editor, this ID is on your custom folder. `ModelAdvisorStandard` is a standard string that you must specify as an input to the `setMetricFamilyParameterValues` method. To visualize results, use the new “Customize Metrics Dashboard Layout” on page 2-3 feature to add custom widgets to the Metrics Dashboard. You can also run the metric engine using APIs to obtain results.

Add Metric Thresholds

In R2018a, you could run the Metrics Dashboard to collect metric data on your model and use the various widgets to explore this data in more detail. Starting in R2018b, you can apply thresholds for categorizing metric data and display these thresholds on the Metrics Dashboard. Setting these thresholds enables you to assess the quality of your model. You use new APIs to specify threshold values corresponding to these three categories:

- Compliant — Metric data that is in an acceptable range.
- Warning — Metric data that requires review.
- Noncompliant — Metric data that requires you to modify your model.

After you specify thresholds and update the active configuration, you can run the Metrics Dashboard. To specify metric thresholds, use these five new classes and two new functions:

Class	Description
<code>slmetric.config.Configuration</code>	Specify metric data categories and custom metric families
<code>slmetric.config.ThresholdConfiguration</code>	Object for holding a collection of metric data thresholds
<code>slmetric.config.Threshold</code>	Specify metric and <code>slmetric.metric.Result</code> property for thresholding
<code>slmetric.config.Classification</code>	Specify categorical metric data ranges
<code>slmetric.metric.ResultClassification</code>	Access metric data threshold values
Function	Description
<code>slmetric.config.getActiveConfiguration</code>	Obtain file path and name of XML file containing active Metrics Dashboard custom configuration
<code>slmetric.config.setActiveConfiguration</code>	Activate custom configuration for metric engine to use

The `slmetric.metric.Result` class contains two new properties: `Category` and `Classifications`. The `slmetric.metric.ResultCollection` class contains the new property `Category`.

Customize Metrics Dashboard Layout

In R2018b, you can customize the layout and functionality of the Metrics Dashboard. For custom metrics, you can visualize metric results by adding widgets to the Metrics Dashboard. You can choose among these visualization techniques:

- Radial gauge
- Single value
- Bar chart
- Distribution heatmap

You can also configure where existing, shipped widgets appear on the Metrics Dashboard or remove these widgets.

To add custom metrics and configure existing metrics to display on the Metrics Dashboard, use these six new classes and two new functions :

Class	Description
<code>slmetric.dashboard.Configuration</code>	Specify custom Metrics Dashboard configuration.
<code>slmetric.dashboard.Layout</code>	Add or remove widgets from the Metrics Dashboard.
<code>slmetric.dashboard.Container</code>	Add widgets next to each other on the Metrics Dashboard.
<code>slmetric.dashboard.Group</code>	Add a title to a group of widgets and place them next to each other on the Metrics Dashboard.
<code>slmetric.dashboard.Widget</code>	Object that holds a Library Reuse, System Interface, or System Info widget. Library Reuse and System Interface widgets are for existing, shipped metrics, so you cannot change the visualization property or metric ID. You cannot assign a metric ID to a System Info widget because it is not a metric.
<code>slmetric.dashboard.CustomWidget</code>	Create widgets in which you specify a visualization property and metric ID.
<code>slmetric.metric.MetaInformation</code>	The <code>slmetric.metric.MetaInformation</code> class properties contain metric metadata. On the Metrics Dashboard, when you click the widget for an individual metric, this metadata is in the table. For custom metrics, when you create a custom metric class, you specify the <code>slmetric.metric.MetaInformation</code> properties. To obtain metric metadata, use the new <code>getMetricMetaInformation</code> method. This method is for the <code>slmetric.Engine</code> object.

Function	Description
<code>slmetric.dashboard.getActiveConfiguration</code>	Return <code>slmetric.dashboard.Configuration</code> object corresponding to active Metrics Dashboard configuration.
<code>slmetric.dashboard.setActiveConfiguration</code>	Set active Metrics Dashboard configuration.

To create widgets for custom metrics, use the `slmetric.dashboard.CustomWidget` class. Add these widgets and shipped widgets in groups and containers or by themselves on the Metrics Dashboard.

For an example of all three of the preceding features, see [Customize Metrics Dashboard Layout and Functionality](#).

Simscape Support with Clone Detection: Detect and refactor clones in Simscape Models

In R2018b, you can run the **Identify Modeling Clones** tool on Simscape models.

JMAAB 4.01 Support: Automate checking of models to comply with JMAAB 4.01 modeling style guidelines

Use these new checks to verify compliance with Japan MATLAB Automotive Advisory Board Checks (JMAAB) guidelines. To execute these checks, open Model Advisor and select **By Task > Modeling Standards for JMAAB**.

Model Advisor Check	Check ID
Check block orientation	mathworks.jmaab.jc_0110
Check usable characters for signal names and bus names	mathworks.jmaab.jc_0222
Check usable characters for parameter names	mathworks.jmaab.jc_0232
Check length of model file name	mathworks.jmaab.jc_0241
Check length of folder name at every level of model path	mathworks.jmaab.jc_0242
Check length of subsystem names	mathworks.jmaab.jc_0243
Check length of Inport and Outport names	mathworks.jmaab.jc_0244
Check length of signal and bus names	mathworks.jmaab.jc_0245
Check length of parameter names	mathworks.jmaab.jc_0246
Check length of block names	mathworks.jmaab.jc_0247
Check if blocks are shaded in the model	mathworks.jmaab.jc_0604

Model Advisor Check	Check ID
Check operator order of Product blocks	mathworks.jmaab.jc_0610
Check icon shape of Logical Operator blocks	mathworks.jmaab.jc_0621
Check for parentheses in Fcn block expressions	mathworks.jmaab.jc_0622
Check usage of Memory and Unit Delay blocks	mathworks.jmaab.jc_0623
Check usage of Lookup Tables	mathworks.jmaab.jc_0626
Check usage of the Saturation blocks	mathworks.jmaab.jc_0628
Check Signed Integer Division Rounding mode	mathworks.jmaab.jc_0642
Check type setting by data objects	mathworks.jmaab.jc_0644
Check if tunable block parameters are defined as named constants	mathworks.jmaab.jc_0645
Check prohibited comparison operation of logical type signals	mathworks.jmaab.jc_0655
Check default/else case in Switch Case blocks and If blocks	mathworks.jmaab.jc_0656
Check usage of Merge block	mathworks.jmaab.jc_0659
Check for unused data in Stateflow Charts	mathworks.jmaab.jc_0700
Check first index of arrays in Stateflow	mathworks.jmaab.jc_0701
Check execution timing for default transition path	mathworks.jmaab.jc_0712
Check for parallel Stateflow state used for grouping	mathworks.jmaab.jc_0721
Check scope of data in parallel states	mathworks.jmaab.jc_0722
Check uniqueness of State names	mathworks.jmaab.jc_0730
Check usage of State names	mathworks.jmaab.jc_0731
Check uniqueness of Stateflow State and Data names	mathworks.jmaab.jc_0732

Model Advisor Check	Check ID
Check repetition of Action types	mathworks.jmaab.jc_0734
Check if each action in state label ends with a semicolon	mathworks.jmaab.jc_0735
Check indentation of Stateflow blocks	mathworks.jmaab.jc_0736
Check uniform spaces before and after operators	mathworks.jmaab.jc_0737
Check comments in state actions	mathworks.jmaab.jc_0738
Check updates to variables used in state transition conditions	mathworks.jmaab.jc_0741
Check boolean operations in condition labels	mathworks.jmaab.jc_0742
Check condition actions in Stateflow transitions	mathworks.jmaab.jc_0743
Check for unexpected backtracking in state transitions	mathworks.jmaab.jc_0751
Check usage of parentheses in Stateflow transitions	mathworks.jmaab.jc_0752
Check condition actions and transition actions in Stateflow	mathworks.jmaab.jc_0753
Check prohibited use of operation expressions in array indices	mathworks.jmaab.jc_0756
Check starting point of internal transition in Stateflow	mathworks.jmaab.jc_0760
Check prohibited combination of state action and flow chart	mathworks.jmaab.jc_0762
Check usage of internal transitions in Stateflow states	mathworks.jmaab.jc_0763
Check usage of transition conditions in Stateflow transitions	mathworks.jmaab.jc_0772

For more information, see Model Checks for Japan MATLAB Automotive Advisory Board (JMAAB) Guideline Compliance.

Additional MAAB 3.0 and High Integrity Checks: Improve quality and compliance to guidelines

- The following table identifies the new and updated checks to verify compliance with MAAB 3.0 guidelines. For information about MAAB guidelines, see Model Advisor Checks for MAAB Guidelines.

Model Advisor Check	Description of Change
Check for blocks not recommended for C/C++ production code deployment	Removed from the Model Advisor By Task > Modeling Standards for MAAB checks. Check is not applicable for verifying compliance with MAAB modeling guidelines.
Check the number of function calls in MATLAB Function blocks	Use this new check to verify compliance with MathWorks® Automotive Advisory Board (MAAB) guidelines. To execute these check, open Model Advisor and select By Task > Modeling Standards for MAAB .

- In R2018b, few of the existing DO-178C/DO-331, and EN 50128, IEC 61508, IEC 62304, and ISO 26262 checks are re-factored to have better usability and consistency in check IDs. For more information, see:
 - Model Checks for DO-178C/DO-331 Standard Compliance
 - Model Checks for IEC 61508, IEC 62304, ISO 26262, and EN 50128 Standard Compliance
- In R2018b, few of the High Integrity Systems Modeling checks have been split in to multiple checks based on the functionality. Also, new checks to support High Integrity Systems Modeling are added. For more information, see
 - Model Checks for DO-178C/DO-331 Standard Compliance
 - Model Checks for IEC 61508, IEC 62304, ISO 26262, and EN 50128 Standard Compliance

High Integrity Systems Modeling Checks: Use the additional conditions to check the configuration parameters

From R2018b, you can use these conditions to check the configuration parameters:

Check Name	Parameter	Parameter Values
Check safety-related diagnostic settings for Stateflow	Undirected event broadcasts	'none' 'warning' (default) 'error' Check passes when value is set to 'error'.
	Transition action specified before condition action	'none' 'warning' (default) 'error' Check passes when value is set to 'error'.
Check safety-related diagnostic settings for sample time	Single task rate transition	'none' (default) 'warning' 'error' Check passes when value is set to 'error'.
	Tasks with equal priority	'none' 'warning' (default) 'error' Check passes when value is set to 'error'.
	Unspecified inheritability of sample time	'none' 'warning' (default) 'error' Check passes when value is set to 'error'.

MISRA C:2012 and Secure Coding Standards: Improve compliance of generated code by using updated Model Advisor checks

Modifications to existing Model Advisor checks that you use to verify compliance with MISRA C:2012 and Secure Coding standards are outlined in this table.

Model Advisor Check	Description of Change
Check configuration parameters for MISRA C:2012 Check configuration parameters for secure coding standards	Checks now analyze the setting for these configuration parameters: <ul style="list-style-type: none"> • External mode • Undirected event broadcasts • Compile-time recursion limit for MATLAB functions • Enable run-time recursion for MATLAB functions
Check for blocks not recommended for MISRA C:2012 Check for blocks not recommended for secure coding standards	Checks now flag the usage of these blocks in a model or subsystem: <ul style="list-style-type: none"> • Compose String • Scan String • String to Double • String to Single • To String

Mnemonic Support: Use keyboard shortcuts with Metrics Dashboard

In R2018b, you can use your keyboard to choose from the actions on the Metric Dashboard toolbar. To enable keyboard shortcuts, on Windows® and Linux® machines, type **Alt + M**. Mac computers do not support keyboard shortcuts.

Check Style for Model Advisor: Create checks that generate interactive reports

In R2018b, the Model Advisor has two new report styles that you can use to view the check results. In addition to Recommended Action, you can now view the results for some of the Model Advisor checks by:

- **Subsystem**. This view organizes the flagged model components by subsystem. The report provides a recommended action for each flagged issue. You can click the hyperlink path to open the affected model component in the model editor.

Check whether block names appear below blocks (recommended check style)

Analysis

Example new style callback (recommended check style)

Run This Check

Result:  Warning

View by Subsystem ▼

Identify blocks where the name is not displayed below the block.

Warning

The following blocks have names that do not display below the blocks:

Subsystem	Block Path
example_sldemo_fuelsys	.../Throttle Angle Fault Switch
example_sldemo_fuelsys/fuel_rate_control	.../fuel_rate_control/validate_sample_time

Recommended Action

Change the location such that the block name is below the block.

Action

Click the button to place block names below blocks


Make block names appear below blocks

- **Block.** This view organizes the flagged model components by block. The report provides a recommended action for each flagged issue. You can click the hyperlink path to open the affected model component in the model editor.

Check whether block names appear below blocks (recommended check style)

Analysis

Example new style callback (recommended check style)

Result:  Warning View by

Issues for block example_sl-demo_fuelsys/Throttle Angle Fault Switch:
Identify blocks where the name is not displayed below the block.

Warning
The following blocks have names that do not display below the blocks:

- [example_sl-demo_fuelsys/Throttle Angle Fault Switch](#)

Recommended Action
Change the location such that the block name is below the block.

Issues for block example_sl-demo_fuelsys/fuel_rate_control/validate_sample_time:
Identify blocks where the name is not displayed below the block.

Warning
The following blocks have names that do not display below the blocks:

- [example_sl-demo_fuelsys/fuel_rate_control/validate_sample_time](#)

Recommended Action
Change the location such that the block name is below the block.

Action

Click the button to place block names below blocks

In R2018b, new report styles are available for the following Model Advisor checks:

- Identify blocks generating inefficient algorithms
- Check state machine type of Stateflow charts
- Check usage of Gain blocks

-
- Check for indexing in blocks
 - Check for prohibited blocks in discrete controllers
 - Check for prohibited sink blocks
 - Check positioning and configuration of ports
 - Check for matching port and signal names
 - Check whether block names appear below blocks
 - Check for mixing basic blocks and subsystems
 - Check for unconnected ports and signal lines
 - Check position of Trigger and Enable blocks
 - Check usage of tunable parameters in blocks
 - Check the display attributes of block names
 - Check display for port blocks
 - Check orientation of Subsystem blocks
 - Check usage of Relational Operator blocks
 - Check use of Simulink in Stateflow charts
 - Check use of default variants
 - Check usage of Discrete-Time Integrator block
 - Check usage of State names
 - Check uniform spaces before and after operators
 - Check comments in state actions
 - Check prohibited comparison operation of logical type signals
 - Check usage of internal transitions in Stateflow states
 - Check usage of transition conditions in Stateflow transitions
 - Check block orientation
 - Check usage of parentheses in Stateflow transitions
 - Check boolean operations in condition labels
 - Check usage of transition conditions in Stateflow transitions
 - Check prohibited use of operation expressions in array indices
 - Check if each action in state label ends with a semicolon
 - Check prohibited combination of state action and flow chart

- Check updates to variables used in state transition conditions
- Check condition actions in Stateflow transitions
- Check starting point of internal transition in Stateflow
- Check usage of Lookup Tables
- Check for parentheses in Fcn block expressions
- Check for blocks not supported for row-major code generation
- Identify TLC S-Functions with unset array layout
- Check input and output datatype for Switch blocks
- Check type setting by data objects
- Check for the Saturation and Saturation Dynamic blocks that perform type casting
- Check usage of fixed-point data type with non-zero bias

To apply the new report formats to your custom Model Advisor checks, use the classes and function listed in this table.

Class	Description
<code>ModelAdvisor.Check</code>	<p>New method <code>setResultDetails</code> associates <code>ResultDetailObjs</code> and <code>ElementResults</code> with the check (<code>CheckObj</code>) in the check callback function.</p> <p>New property <code>ModelAdvisor.Check.ResultDetails</code> stores the <code>ResultDetailObjs</code>.</p> <p>In the <code>setCallbackFcn</code> method, input argument <code>'DetailStyle'</code> specifies the callback function for the new report style. Define the new <code>'DetailStyle'</code> callback function type in the <code>ModelAdvisor.Check.CallbackStyle</code> property.</p>
<code>ModelAdvisor.ResultDetail</code>	<p>Defines the result detail objects for a specific check object. Each object stores the result details for one model element, such as a block that violates a check.</p>

Functionality Being Removed or Changed

Instances of `slmetric.metric.Result` for the `mathworks.metrics.CloneContent` and `mathworks.metrics.LibraryContent` metrics contain these differences between R2018a and R2018b:

- Previously, for the `mathworks.metrics.CloneContent` metric, the `Value` property provided the number of components in a clone. The `Measures` property was not applicable. In R2018b, the `Value` property provides the fraction of the total number of subcomponents that are clones. The `Measures` property is a vector containing the number of clones, total number of components, and the clone group number.
- Previously, for the `mathworks.metrics.LibraryContent` metric, the `Value` property provided the number of components involved in a library, excluding clones. The `Measures` property was not applicable. In R2018b, the `Value` property provides the fraction of the total number of components that are linked-library blocks. The `Measures` property is a vector containing the number of linked-library blocks and total number of components.

The Metrics Dashboard incorporates these changes. Previously, the **Library Reuse** widget displayed percentages that were a combination of five metrics. The widget directly used the Clone detection (`mathworks.metrics.CloneDetection`) and Library link (`mathworks.metrics.LibraryCount`) metrics. To calculate percentages, the widget indirectly used the MATLAB Function count (`mathworks.metrics.MatlabFunctionCount`), the Chart count (`mathworks.metrics.StateflowChartCount`), and the Subsystem count (`mathworks.metrics.SubSystemCount`) metrics.

In R2018b, The Library Reuse widget is **Potential Reuse** and **Actual Reuse** bars. The **Potential Reuse** bar displays the `mathworks.metrics.CloneContent` `Value` as a percentage. The **Actual Reuse** bar displays the `mathworks.metrics.LibraryContent` `Value` as a percentage.

For the `mathworks.metrics.CloneContent` and `mathworks.metrics.LibraryCount` metrics, changing the `slmetric.metric.Result` `Measures` and `Values` properties supports displaying metric threshold values on the Metrics Dashboard because you specify metric thresholds on a single metric and not on a combination of metrics. Metric thresholds is a new R2018b feature. For more information, see [Customize Metrics Dashboard Layout and Functionality](#).

R2018a

Version: 4.1

New Features

Bug Fixes

Additional Checks for MAAB 3.0 and JMAAB 4.0 Guidelines: Automate checking for MAAB 3.0 guidelines for Simulink, Stateflow, Variant Subsystems, and MATLAB Function Blocks and JMAAB 4.0 guidelines

MAAB Modeling Checks

Use these new checks to verify compliance with MathWorks® Automotive Advisory Board (MAAB) guidelines. To execute these checks, open Model Advisor (Simulink) and select **By Task > Modeling Standards for MAAB**.

For information about MAAB® guidelines, see Model Advisor Checks for MAAB Guidelines (Simulink).

By Task > Modeling Standards for MAAB subfolder	Model Advisor Check	Addresses Guideline
Naming Convention	Check Simulink bus signal names	na_0030: Usable characters for Simulink Bus names
Model Architecture	Check unused ports in Variant Subsystems	na_0020: Number of inputs to variant subsystems
Model Architecture	Check use of default variants	na_0036: Default variant
Model Architecture	Check use of single variable variant conditionals	na_0037: Use of single variable variant conditionals
Model Architecture	Check number of Stateflow states per container	na_0040: Number of states per container
Simulink	Check fundamental logical and numerical operations	na_0002: Appropriate implementation of fundamental logical and numerical operations
Simulink	Check usage of merge blocks	na_0032: Use of merge blocks

By Task > Modeling Standards for MAAB subfolder	Model Advisor Check	Addresses Guideline
Simulink	Check logical expressions in 'If' blocks	na_0003: Simple logical expressions in If Condition block
Stateflow	Check nested states in Stateflow charts	na_0038: Levels in Stateflow charts
Stateflow	Check use of Simulink in Stateflow charts	na_0039: Use of Simulink in Stateflow charts
MATLAB Functions	Check usage of reserved keywords in Simulink	na_0019: Restricted Variable Names
MATLAB Functions	Check usage of character vector inside MATLAB Function block	na_0021: Strings
MATLAB Functions	Check Recommended patterns for Switch/Case Statements	na_0022: Recommended patterns for Switch/Case statements

Modifications to existing MAAB checks are outlined in this table.

Model Advisor Check	Description of Change
Check entry formatting in State blocks in Stateflow charts	These checks now require a Stateflow® license. These checks are included in the Model Advisor interface only when a Stateflow license is detected.
Check for mismatches between names of Stateflow ports and associated signals	
Check for Strong Data Typing with Simulink I/O	
Check for indexing in blocks	
Check for MATLAB expressions in Stateflow charts	
Check transition orientations in flow charts	
Check usage of exclusive and default states in state machines	
Check transition actions in Stateflow charts	
Check for unary minus operations on unsigned integers in Stateflow charts	
Check for equality operations between floating-point expressions in Stateflow charts	
Check return value assignments of graphical functions in Stateflow charts	
Check usage of return values from a graphical function in Stateflow charts	
Check default transition placement in Stateflow charts	

Model Advisor Check	Description of Change
Check for pointers in Stateflow charts Check for event broadcasts in Stateflow charts Check for bitwise operations in Stateflow charts Check for comparison operations in Stateflow charts	
Check the display attributes of block names	You can now use the input parameters in the Model Advisor Configuration Editor to customize the blocks and masks to be analyzed the check.
Check position of Trigger and Enable blocks	Check now verifies that For Each, For Iterator, and While Iterator blocks are in a uniform location on the subsystem diagram.

JMAAB Modeling Checks

Checks that verify compliance with Japan MATLAB Automotive Advisory Board (JMAAB) guidelines are now available in the Model Advisor under the new menu item **By Task > Modeling Standards for JMAAB**.

The following table identifies new checks to verify compliance with JMAAB 4.0 guidelines. For information about JMAAB guidelines, see Model Advisor Checks for MAAB Guidelines (Simulink).

By Task > Modeling Standards for JMAAB Subfolder	Model Advisor Check	Addresses Guideline
Simulink	Check usage of Discrete-Time Integrator block	jc_0627: Guideline for using the Discrete-Time Integrator block
Simulink	Check for blocks with a fixed-point data type whose bias is not zero	jc_0643: Fixed-point setting

By Task > Modeling Standards for JMAAB Subfolder	Model Advisor Check	Addresses Guideline
Simulink	Check input and output datatype for Switch blocks	jc_0650: Block input/output data type with switching function
Simulink	Check input signal data types in product blocks that perform division	jc_0611: Input signal sign during product block division

The following table identifies MAAB checks are also applicable to JMAAB 4.0 guidelines. These checks are available in the Model Advisor (Simulink) under **By Task > Modeling Standards for JMAAB**. There are no changes to the check IDs.

By Task > Modeling Standards for JMAAB Subfolder	Model Advisor Check
Naming Conventions	Check file names
	Check folder names
	Check subsystem names
	Check port block names
	Check character usage in signal labels
	Check character usage in block names
Model Architecture	Check for mixing basic blocks and subsystems
Model Configuration Options	Check Implement logic signals as Boolean data (vs. double)
Simulink	Check for Simulink diagrams using nonstandard display attributes
	Check font formatting
	Check positioning and configuration of ports
	Check whether block names appear below blocks

By Task > Modeling Standards for JMAAB Subfolder	Model Advisor Check
	Check the display attributes of block names
	Check position of Trigger and Enable blocks
	Check for nondefault block attributes
	Check Trigger and Enable block names
	Check signal line labels
	Check for propagated signal labels
	Check for unconnected ports and signal lines
	Check for prohibited blocks in discrete controllers
	Check for prohibited sink blocks
	Check usage of Switch blocks
	Check usage of Relational Operator blocks
	Check for indexing in blocks
	Check usage of tunable parameters in blocks
	Check orientation of Subsystem blocks
Stateflow	Check transition orientations in flow charts
	Check return value assignments of graphical functions in Stateflow charts
	Check default transition placement in Stateflow charts
	Check for Strong Data Typing with Simulink I/O
	Check Stateflow data objects with local scope
	Check usage of return values from a graphical function in Stateflow charts
	Check for MATLAB expressions in Stateflow charts
	Check for pointers in Stateflow charts
	Check for event broadcasts in Stateflow charts
	Check transition actions in Stateflow charts

By Task > Modeling Standards for JMAAB Subfolder	Model Advisor Check
	Check for bitwise operations in Stateflow charts
	Check for unary minus operations on unsigned integers in Stateflow charts
	Check for comparison operations in Stateflow charts
	Check for mismatches between names of Stateflow ports and associated signals
MATLAB Function	Check input and output settings of MATLAB Functions
	Check MATLAB Function metrics
	Check MATLAB code for global variables

Block Constraint Authoring with Edit-Time: Define checks for supported or unsupported blocks and parameters while editing

In R2018a, there are seven new classes and two new functions that you can use to create block and parameter constraints. You can use the `sl_customization` function template to create basic Model Advisor checks. These checks include these constraints and a check algorithm callback. You can check your model as you edit or run the checks interactively after you complete your model design.

The new classes and their descriptions are in the table.

Class	Description
<code>Advisor.authoring.PositiveBlockParameterConstraint</code>	Check for supported block parameter values.
<code>Advisor.authoring.NegativeBlockParameterConstraint</code>	Check for unsupported block parameter values.
<code>Advisor.authoring.PositiveModelParameterConstraint</code>	Check for supported model parameter values.

Class	Description
Advisor.authoring.NegativeModelParameterConstraint	Check for unsupported model parameter values.
Advisor.authoring.PositiveBlockTypeConstraint	Check for supported blocks.
Advisor.authoring.NegativeBlockTypeConstraint	Check for unsupported blocks.
Advisor.authoring.CompositeConstraint	Check whether blocks or parameters meet a combination of constraints.

The new functions and their descriptions are in the table.

Function	Description
Advisor.authoring.generateBlockConstraintsDataFile	Generate XML data file for custom check for block constraints.
Advisor.authoring.createBlockConstraintCheck	Create Model Advisor check for registering block constraints.

You can also create constraints that check for pre-requisite constraints before checking the actual constraint. For more information, see Define Checks for Supported or Unsupported Blocks and Parameters.

Clone Refactoring Workflow: Apply multiple refactoring steps to the same model

In R2017b, for each refactoring step, the Identify Modeling Clones tool created a new model. In R2018a, you can apply multiple refactoring steps to the same model. For each step, the tool creates a back-up model. The back-up models are in the folder that has the prefix m2m_ plus the model name. For a single model, this enhanced functionality makes it easier for you to replace clones with links to library blocks. For more information, see Enable Component Reuse by Using Clone Detection.

Automatic Refactoring for Similar Clones: Add masks to similar clones and refactor model

In R2017b, you could use the Identify Modeling Clones tool to identify exact clones and replace them with links to library blocks. The tool also identified similar clones (that is,

clones that had identical block types and connections but different parameter settings or values). The tool identified similar and exact clones as part of different steps.

In R2018a, for similar clones, the Identify Modeling Clones tool creates a masked library subsystem. The refactored model contains links from the clone instances to this masked library subsystem. The tool identifies similar and exact clones as part of the same steps. You specify which clones you want to detect by setting the value of the **Maximum number of different parameters** parameter. A value of 0 indicates that you want the tool to identify only exact clones.

Replacing clones with links to library blocks enables component reuse. If you have Simulink Coder or Embedded Coder software, you can generate reusable code for library subsystems. For more information, see [Enable Component Reuse by Using Clone Detection](#).

Clone Detection Exclusion Editor: Exclude subsystems and referenced models from clone detection

In R2018a, there is a new Clone Detection Exclusion Editor that you can use to exclude a subsystem or referenced model from the Identify Modeling Clones tool. For subsystems, right-click the subsystem and select **Identify Modeling Clones > Subsystem and Its Contents > Add to exclusions**. For referenced models, right-click the Model block and select **Identify Modeling Clones > Model Reference > Add to exclusions**. You can use the Exclusion Editor to specify a rationale for excluding subsystems and referenced models and whether to store exclusions in a model file. For more information, see [Enable Component Reuse by Using Clone Detection](#).

Automatic Data Store Memory Block Elimination: Identify and refactor Data Store Memory Block blocks with Model Transformer

In R2018a, use the Model Transformer tool to refactor a model to eliminate Data Store Memory, Data Store Read, and Data Store Write blocks. Eliminating these blocks improves model readability by making data-dependency explicit. If you have Simulink Coder™, eliminating these blocks may improve the efficiency of the generated code by reducing the number of global variables, the corresponding reads and writes to these global variables, and stack size. For more information, see [Improve Model Readability by Eliminating Local Data Store Blocks](#).

Grid Visualization for Metrics: View results of Model Advisor checks in a grid to identify patterns in results

In R2017b, after collecting metric data by using the Metric Dashboard, you could view results for High Integrity and MAAB Compliance metrics in tabular format. You viewed this data by clicking the widgets in the **MODELING GUIDELINE COMPLIANCE** section.

In R2018a, when you click the **High Integrity Compliance** and **MAAB Compliance** widgets, you can view results in a table or a grid. In the toolbar, you change views by clicking **Table** or **Grid**.

Viewing results in a grid enables you to identify compliance check issues and failure patterns quickly. The grid contains a row for each model component and a column for each check. The colors in each grid cell indicate this status.

Color	Check Status
Red	Fail
Orange	Warning
Green	Pass
Gray	Not run

The colors in the row and column headers indicate the worst status for that component or check. For example, for a row, if the worst status is a failure, the row header is red.

Placing your cursor over a cell displays the component, check status, and check name. You can click individual cells to navigate to the corresponding block and identify compliance issues for that block. To navigate to the corresponding check in the Model Advisor, click a column header. To navigate to the corresponding model component, click a row header. For more information, see *Collect and Explore Metric Data by Using the Metrics Dashboard*.

MathWorks High-Integrity Guidelines and Checks: Verify compliance with safety standards by using high-integrity checks and guidelines

High-Integrity System Modeling Checks

This table identifies modifications to existing high-integrity system modeling checks.

Model Advisor Check	Description of Change
<p>DO-178C/DO-331 checks:</p> <ul style="list-style-type: none"> • Check usage of shift operations for Stateflow data • Check assignment operations in Stateflow Charts • Check state machine type of Stateflow charts • Check Stateflow charts for ordering of states and transitions • Check Stateflow debugging options • Check for inconsistent vector indexing methods • Check Stateflow charts for strong data typing • Check Stateflow charts for unary operators • Check Stateflow charts for uniquely defined data objects <p>IEC 61508, IEC 62304, EN 50128, and ISO 26262 checks:</p> <ul style="list-style-type: none"> • Check usage of shift operations for Stateflow data • Check assignment operations in Stateflow Charts • Check state machine type of Stateflow charts • Check Stateflow charts for ordering of states and transitions • Check Stateflow debugging options • Check for inconsistent vector indexing methods 	<p>Checks require a Stateflow license. These checks are available in the Model Advisor only when a Stateflow license is detected.</p>

Model Advisor Check	Description of Change
<ul style="list-style-type: none"> • Check Stateflow charts for strong data typing • Check Stateflow charts for unary operators • Check Stateflow charts for uniquely defined data objects • Display model metrics and complexity report 	
<p>DO-178C/DO-331: Check for model elements that do not link to requirements</p> <p>IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check for model elements that do not link to requirements</p>	<p>Checks require a Simulink Requirements™ license. These checks are available in the Model Advisor only when a Simulink Requirements license is detected.</p>
<p>Check for blocks not recommended for MISRA C:2012</p> <p>Check for blocks not recommended for MISRA C:2012</p>	<p>Checks analysis now includes the content in library linked blocks and masked subsystems.</p>
<p>Check for blocks not recommended for C/C++ production code deployment</p> <p>Check for blocks not recommended for C/C++ production code deployment</p>	<p>Checks analysis now includes the content in library linked blocks.</p>

High-Integrity Modeling Guidelines

These high-integrity system modeling guidelines are introduced in R2018a:

- hisl_0056: Configuration Parameters > Optimization > Optimize using the specified minimum and maximum values
- hisl_0066: Usage of Gain blocks
- hisl_0314: Configuration Parameters > Diagnostics > Data Validity > Signals
- hisf_0016: Stateflow port names
- hisf_0017: Stateflow data object scoping

This table identifies removed and modified high-integrity system modeling guidelines. For a complete list of high-integrity system modeling guidelines, including their applicable Model Advisor checks, see Model Advisor Checks for High-Integrity Modeling Guidelines (Simulink).

High-Integrity Modeling Guideline	Rationale
hisl_0202: Use of data conversion blocks to improve MISRA C:2012 compliance	Removed - guideline no longer applies. The code generation process does not produce this MISRA violation in the generated code.
hisl_0401: Encapsulation of code to improve MISRA C:2012 compliance	Removed - not a modeling guideline. The guideline addresses issues in source code external to a model. Embedded Coder® does not directly call assembly language code.
hisl_0402: Use of custom #pragma to improve MISRA C:2012 compliance	Removed - not a modeling guideline. The guideline addresses issues in source code external to a model.
hisl_0403: Use of char data type to improve MISRA C:2012 compliance	Removed - not a modeling guideline. The guideline addresses issues in source code external to a model. Embedded Coder does not directly create data of type char.
hisl_0055: Prioritization of code generation objectives for high-integrity systems	Removed - guideline is redundant. Model configuration parameter considerations are covered by existing high-integrity systems guidelines.
hisl_0042: Configuration Parameters > Solver > Tasking and sample time options	<p>Removed criteria A, "Periodic sample time constraint to specified and assign values to Sample time properties". Criteria A no longer applies because there is no known safety issue with the periodic sample time constraint values.</p> <p>Change affects checks:</p> <ul style="list-style-type: none"> • DO-178C/DO-331: Check safety-related solver settings for tasking and sample-time • EC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related solver settings for tasking and sample-time

High-Integrity Modeling Guideline	Rationale
hisl_0310: Configuration Parameters > Diagnostics > Model Referencing	Set configuration parameter Model block version mismatch to none . This change reflects behavior of the corresponding Model Advisor check.
hisl_0016: Usage of blocks that compute relational operators	Add the If block to the list of blocks that compute relational operations. This change reflects behavior of Model Advisor check.
hisl_0017: Usage of blocks that compute relational operators (2)	Add Rational B, "For Relational Operator blocks, ensure that all input signals are of the same data type". This change reflects behavior of the corresponding Model Advisor check.
<p>hisl_0046: Configuration Parameters > Code Generation > Optimization > Block reduction</p> <p>hisl_0051: Configuration Parameters > Code Generation > Optimization > Loop unrolling threshold</p> <p>hisl_0052: Configuration Parameters > Code Generation > Optimization > Data initialization</p> <p>hisl_0053: Configuration Parameters > Code Generation > Optimization > Remove code from floating-point to integer conversions that wraps out-of-range values</p> <p>hisl_0054: Configuration Parameters > Code Generation > Optimization > Remove code that protects against division arithmetic exceptions</p>	<p>In R2018a, the Optimization pane in the Configuration Parameters dialog box moved to Code Generation > Optimization. The title of and support documentation for the affected guidelines were updated to reflect this change. Where applicable, the Model Advisor checks were updated to reflect this change.</p>

High-Integrity Modeling Guideline	Rationale
hisl_0045: Configuration Parameters > Code Generation > Optimization > Implement logic signals as Boolean data (vs. double)	In R2018a, configuration parameter Implement logic signals as Boolean data (vs. double) moved to the new Math and Data Types pane in the Configuration Parameters dialog box. The title of and support documentation for the guideline were updated to reflect this change.
hisl_0019: Usage of Bitwise Operator block	Remove Criteria B, "Choose an output data type that represents zero exactly" from the guideline. Criteria B does not apply because the Bitwise Operator block, by design, accepts only signed or unsigned integer to produce the output data type that represents zero exactly. It does not accept other data types.
hisl_0102: Data type of loop control variables to improve MISRA C:2012 compliance	Remove While loops and While Iterator blocks. These are not recommended for safety critical systems because an infinite loop can occur. Guideline does not have corresponding checks.
hisl_0031: File and folder names	Remove folder naming recommendations because there is no known safety issue with folder names. Guideline title was updated accordingly.

MISRA C: 2012 Modeling Checks: Improve compliance of generated code by using MISRA C:2012 standards checks

Use this new check to verify compliance of your generated code with MISRA C:2012 standards. To execute this check, open Model Advisor (Simulink) and select **By Task > Modeling Standards for MISRA C:2012**

Model Advisor Check	Description	Addresses Standards
Check bus object names that are used as element names	Check now identifies Simulink.Bus object names that are used as Simulink.Bus element names.	<ul style="list-style-type: none"> • MISRA C:2012 Rule 5.6 • MISRA AC AGC Rule 5.3

Modifications to existing compliance checks are outlined in this table.

Model Advisor Check	Description of Change
Check for bitwise operations on signed integers	The check assumes that code is generated for the whole model. When code is generated by a subsystem build or export functions, the check can produce incorrect results.
Check for blocks not recommended for MISRA C:2012	Checks analysis now includes the content in library linked blocks and masked subsystems.
Check for blocks not recommended for C/C++ production code deployment	

For information about MISRA C[®] versions and updates, see MISRA C Guidelines (Embedded Coder).

Secure Coding Modeling Checks: Update to Secure Coding compliance checks

Modifications to existing secure coding checks are outlined in this table.

Model Advisor Check	Description of Change
Check for bitwise operations on signed integers	Check now identifies blocks in the top model only.
Check for blocks not recommended for C/C++ production code deployment	Checks analysis now includes the content in library linked blocks and masked subsystems.

For information about MISRA C versions and updates, see MISRA C Guidelines (Embedded Coder).

Enhanced Edit-Time Checking Support: Edit-time checking for blocks not recommended for C/C++ production code deployment

In R2017b, you could run the Model Advisor check `mathworks.codegen.PCGSupport` to identify blocks that were not supported by code generation or were not recommended for C/C++ production code deployment. In R2018a, you can use edit-time checking to identify these blocks earlier on in the design process. For more information, see [Check for Compliance Using the Model Advisor and Edit-Time Checking](#).

Model Advisor Support for Inactive Variants: Run Model Advisor checks on active and inactive variants and generate report

In R2017b, you could run Model Advisor checks only on the active variant of a model. You had to manually activate the various variant choices to run the Model Advisor on different variants.

In R2018a, you can run Model Advisor checks on valid variant configurations. Use the Variant Manager to define these configurations. Set the new `Advisor.Application` class property `AnalyzeVariants` to `true`. The Model Advisor generates a separate HTML report of check results for each variant configuration.

Metric Engine Improvement: Collect and analyze metric data faster

In R2018a, for a given analysis, the metric engine collects and analyzes metric data faster than in R2017b. Also, when you open the Metric Dashboard for a model in which you previously generated results, the results are now loaded more quickly. For more information, see [Collect Model Metrics](#).

R2017b

Version: 4.0

New Features

Compatibility Considerations

Simulink Verification and Validation Packaging: Moved compliance checking, model metrics, clone detection and refactoring, edit-time checking and model transformer to Simulink Check

As of R2017b, Simulink Verification and Validation™ transitions to three new products, Simulink Requirements, Simulink Coverage, and Simulink Check.

- Requirements traceability and Requirements Management Interface (RMI) functionality have moved to the Simulink Requirements product.
- Model and generated code coverage functionality, and component verification functions such as `slvnmmakeharness`, have moved to the Simulink Coverage™ product.
- Compliance checking, model metrics, clone detection and refactoring, and model transformer functionality have moved to the Simulink Check product.

Metrics Dashboard: Collect and view metric data for quality assessment

The Metrics Dashboard collects and integrates quality metric data from multiple Model-Based Design tools to provide you with an assessment of your project quality status. In R2017b, by using the dashboard, you can collect and explore metric data for:

- Model size
- Modeling guidelines compliance
- Model componentization and clone detection

To explore the data in more detail, click an individual metric. For your selected metric, a table displays the value, aggregated value, and measures (if applicable) at the model component level. From the table, the dashboard provides traceability and hyperlinks to the data source so that you can get detailed results and recommended actions for troubleshooting issues.

Open the Metrics Dashboard from the model editor window by selecting **Analysis > Metrics Dashboard**. Or, at the command line, enter `metricsdashboard(system)`.

For more information, see *Collect and Explore Metric Data by Using the Metrics Dashboard*.

MathWorks High-Integrity Guidelines and Checks: Verify compliance with safety standards by using high-integrity checks and guidelines

Categorization of the Model Advisor Checks for High-Integrity Systems

You can use the Model Advisor to check compliance with safety standards by using the high-integrity checks. To execute these checks, Open the Model Advisor (Simulink) and select the safety standard:

- **By Task > Modeling Standards for DO-178/DO-331 > High-Integrity Systems**
- **By Task > Modeling Standards for EN 50128 > High-Integrity Systems**
- **By Task > Modeling Standards for IEC 61508 > High-Integrity Systems**
- **By Task > Modeling Standards for IEC 62304 > High-Integrity Systems**
- **By Task > Modeling Standards for ISO 26262 > High-Integrity Systems**

The high-integrity checks are categorized into these subgroups:

- Simulink
- Stateflow
- MATLAB
- Configuration
- Requirements
- Code

High-Integrity Model Advisor Checks for DO-178C/DO-331 Standards

The following table identifies the Model Advisor checks that have been introduced in R2017b to check compliance with safety standards DO-178C/DO-331.

These checks are available at **By Task > Modeling Standards for DO-178/DO-331 > High-Integrity Systems**. The high-integrity subgroup in which the check resides is defined in the table.

High-Integrity Systems Subgroup	Check Name
Simulink	Check for root Inports with missing properties

High-Integrity Systems Subgroup	Check Name
Simulink	Check for root Inports with missing range definitions
Stateflow	Check Stateflow charts for transition paths that cross parallel state boundaries
Stateflow	Check Stateflow charts for strong data typing
Stateflow	Check usage of shift operations for Stateflow data
Stateflow	Check assignment operations in Stateflow Charts
Stateflow	Check Stateflow charts for unary operators
Stateflow	Check usage of Stateflow constructs
Configuration	Check safety-related solver settings for simulation time
Configuration	Check safety-related solver settings for solver options
Configuration	Check safety-related solver settings for tasking and sample-time
Configuration	Check safety-related diagnostic settings for Merge blocks
Configuration	Check safety-related diagnostic settings for Stateflow
Configuration	Check safety-related optimization settings for Loop unrolling threshold
Code	Check for blocks not recommended for MISRA C:2012
Code	Check configuration parameters for MISRA C:2012

The following table identifies modifications to existing Model Advisor checks for DO-178C/DO-331 safety standards.

Model Advisor Check	Description of Change
Check for model elements that do not link to requirements	Check title has been updated. In previous releases, the title of this check was Check for blocks that do not link to requirements . The check ID did not change.

Model Advisor Check	Description of Change
Check model for block upgrade issues	No longer available as a Modeling Standards for DO-178C/DO-331 check. For more information, see "DO-178C/DO-331 Modeling Checks: Removed Model Advisor check "Check model for block upgrade issues"" on page 4-20.

High-Integrity Model Advisor Checks for EN 50128, IEC 61508, IEC 62304, and ISO 26262 Standards

The following table identifies the Model Advisor checks that have been introduced in R2017b to check compliance with safety standards EN 50128, IEC 61508, IEC 62304, and ISO 26262.

These checks are available at:

- **By Task > Modeling Standards for EN 50128 > High-Integrity Systems**
- **By Task > Modeling Standards for IEC 61508 > High-Integrity Systems**
- **By Task > Modeling Standards for IEC 62304 > High-Integrity Systems**
- **By Task > Modeling Standards for ISO 26262 > High-Integrity Systems**

The high-integrity subgroup in which the check resides is defined in the table.

High-Integrity Systems Subgroup	Check Name
Simulink	Check usage of lookup table blocks
Simulink	Check for blocks not recommended for C/C++ production code deployment
Simulink	Check for variant blocks with 'Generate preprocessor conditionals' active
Simulink	Check usage of Signal Routing blocks
Stateflow	Check Stateflow charts for transition paths that cross parallel state boundaries
Stateflow	Check Stateflow charts for ordering of states and transitions

High-Integrity Systems Subgroup	Check Name
Stateflow	Check Stateflow debugging options
Stateflow	Check Stateflow charts for uniquely defined data objects
Stateflow	Check Stateflow charts for strong data typing
Stateflow	Check usage of shift operations for Stateflow data
Stateflow	Check assignment operations in Stateflow Charts
Stateflow	Check Stateflow charts for unary operators
Stateflow	Check usage of Stateflow constructs
Configuration	Check safety-related optimization settings
Configuration	Check safety-related model referencing settings
Configuration	Check safety-related code generation settings
Configuration	Check safety-related diagnostic settings for solvers
Configuration	Check safety-related solver settings for simulation time
Configuration	Check safety-related solver settings for solver options
Configuration	Check safety-related solver settings for tasking and sample-time
Configuration	Check safety-related diagnostic settings for sample time
Configuration	Check safety-related diagnostic settings for signal data
Configuration	Check safety-related diagnostic settings for parameters
Configuration	Check safety-related diagnostic settings for data used for debugging
Configuration	Check safety-related diagnostic settings for data store memory
Configuration	Check safety-related diagnostic settings for type conversions
Configuration	Check safety-related diagnostic settings for signal connectivity
Configuration	Check safety-related diagnostic settings for bus connectivity

High-Integrity Systems Subgroup	Check Name
Configuration	Check safety-related diagnostic settings that apply to function-call connectivity
Configuration	Check safety-related diagnostic settings for compatibility
Configuration	Check safety-related diagnostic settings for model initialization
Configuration	Check safety-related diagnostic settings for model referencing
Configuration	Check safety-related diagnostic settings for saving
Configuration	Check safety-related diagnostic settings for Merge blocks
Configuration	Check safety-related diagnostic settings for Stateflow
Configuration	Check safety-related optimization settings for Loop unrolling threshold
Requirements	Check for model elements that do not link to requirements
Code	Check configuration parameters for MISRA C:2012
Code	Check for blocks not recommended for MISRA C:2012

High-Integrity Modeling Guidelines

High-integrity system modeling guideline hisl_0070: Placement of requirement links in a model was introduced in R2017b.

These high-integrity system modeling guidelines were removed in R2017b:

- hisf_0010: Usage of transition paths (looping out of parent of source and destination objects)
- hisf_0012: Chart comments

The high-integrity system modeling guidelines in this table were updated to include new Model Advisor checks for DO-178C/DO-331, EN 50128, IEC 61508, IEC 62304, and ISO 26262 safety standards. Where applicable, the table also identifies additional modifications.

For a complete list of high-integrity system modeling guidelines, including their applicable Model Advisor checks, see Model Advisor Checks for High-Integrity Modeling Guidelines (Simulink).

High-Integrity Modeling Guideline	Description of Change
hisl_0002: Usage of Math Function blocks (rem and reciprocal)	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check usage of Math Operations blocks
hisl_0004: Usage of Math Function blocks (natural logarithm and base 10 logarithm)	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check usage of Math Operations blocks
hisl_0005: Usage of Product blocks	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for signal data
hisl_0013: Usage of data store blocks	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for data store memory
hisl_0018: Usage of Logical Operator block	Removed Model Advisor check Check safety-related optimization settings as it is covered via the prerequisite guideline.

High-Integrity Modeling Guideline	Description of Change
hisl_0020: Blocks not recommended for MISRA C:2012 compliance	<p>New Model Advisor checks:</p> <ul style="list-style-type: none"> • DO-178C/DO-331: Check for blocks not recommended for MISRA C:2012 • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check for blocks not recommended for MISRA C:2012 • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check for blocks not recommended for C/C++ production code deployment <p>Added:</p> <ul style="list-style-type: none"> • Added From Workspace and S-Function Builder blocks to the list of blocks not recommended for MISRA compliance • Identified the deprecated Lookup Table blocks (Lookup and Lookup2D).
hisl_0022: Data type selection for index signals	Removed n-D Lookup Table (internal type index selection) from the list of blocks that use a signal index.
hisl_0023: Verification of model and subsystem variants	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check for variant blocks with 'Generate preprocessor conditionals' active
hisl_0024: Inport interface definition	<p>New Model Advisor check for DO-178C/DO-331: Check for root Inports with missing properties</p> <p>Updated guideline description to include Simulink signal object that explicitly resolves to the connected signal line.</p>
hisl_0025: Design min/max specification of input interfaces	New Model Advisor check for DO-178C/DO-331: Check for root Inports with missing range definitions
hisl_0026: Design min/max specification of output interfaces	New Model Advisor check for DO-178C/DO-331: Check for root Outports with missing range definitions

High-Integrity Modeling Guideline	Description of Change
hisl_0033: Usage of Lookup Table blocks	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check usage of lookup table blocks
hisl_0034: Usage of Signal Routing blocks	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check usage of Signal Routing blocks
hisl_0036: Configuration Parameters > Diagnostics > Saving	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for saving
hisl_0037: Configuration Parameters > Model Referencing	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related model referencing settings
hisl_0038: Configuration Parameters > Code Generation > Comments	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related code generation settings
hisl_0039: Configuration Parameters > Code Generation > Interface	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related code generation settings
hisl_0040: Configuration Parameters > Solver > Simulation time	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check safety-related solver settings for simulation time • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related solver settings for simulation time
hisl_0041: Configuration Parameters > Solver > Solver options	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check safety-related solver settings for solver options • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related solver settings for solver options

High-Integrity Modeling Guideline	Description of Change
hisl_0042: Configuration Parameters > Solver > Tasking and sample time options	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check safety-related solver settings for tasking and sample-time • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related solver settings for tasking and sample-time
hisl_0043: Configuration Parameters > Diagnostics > Solver	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for solvers
hisl_0044: Configuration Parameters > Diagnostics > Sample Time	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for sample time
hisl_0045: Configuration Parameters > Optimization > Implement logic signals as Boolean data (vs. double)	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related optimization settings
hisl_0046: Configuration Parameters > Optimization > Block reduction	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related optimization settings
hisl_0047: Configuration Parameters > Code Generation > Code Style	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related code generation settings
hisl_0048: Configuration Parameters > Optimization > Application lifespan (days)	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related optimization settings
hisl_0049: Configuration Parameters > Code Generation > Symbols	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related code generation settings

High-Integrity Modeling Guideline	Description of Change
hisl_0051: Configuration Parameters > Optimization > Signals and Parameters > Loop unrolling threshold	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check safety-related optimization settings for Loop unrolling threshold • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related optimization settings for Loop unrolling threshold
hisl_0052: Configuration Parameters > Optimization > Data initialization	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related optimization settings
hisl_0053: Configuration Parameters > Optimization > Remove code from floating-point to integer conversions that wraps out-of-range values	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related optimization settings
hisl_0054: Configuration Parameters > Optimization > Remove code that protects against division arithmetic exceptions	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related optimization settings
hisl_0060: Configuration parameters that improve MISRA C:2012 compliance	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check configuration parameters for MISRA C:2012 • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check configuration parameters for MISRA C:2012
hisl_0061: Unique identifiers for clarity	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check usage of Stateflow constructs • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check Stateflow charts for uniquely defined data objects

High-Integrity Modeling Guideline	Description of Change
hisl_0301: Configuration Parameters > Diagnostics > Compatibility	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for compatibility
hisl_0302: Configuration Parameters > Diagnostics > Data Validity > Parameters	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for parameters
hisl_0303: Configuration Parameters > Diagnostics > Merge block	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check safety-related diagnostic settings for Merge blocks • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for Merge blocks
hisl_0304: Configuration Parameters > Diagnostics > Model initialization	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for model initialization
hisl_0305: Configuration Parameters > Diagnostics > Debugging	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for data used for debugging
hisl_0306: Configuration Parameters > Diagnostics > Connectivity > Signals	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for signal connectivity
hisl_0307: Configuration Parameters > Diagnostics > Connectivity > Buses	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for bus connectivity
hisl_0308: Configuration Parameters > Diagnostics > Connectivity > Function calls	New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings that apply to function-call connectivity

High-Integrity Modeling Guideline	Description of Change
hisl_0309: Configuration Parameters > Diagnostics > Type Conversion	<p>New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for type conversions</p> <p>Added Type Conversion parameters:</p> <ul style="list-style-type: none"> • Unnecessary type conversion • 32-bit integer to single precision float conversion
hisl_0310: Configuration Parameters > Diagnostics > Model Referencing	<p>New Model Advisor check for IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for model referencing</p>
hisl_0311: Configuration Parameters > Diagnostics > Stateflow	<p>New Model Advisor checks:</p> <ul style="list-style-type: none"> • DO-178C/DO-331: Check safety-related diagnostic settings for Stateflow • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check safety-related diagnostic settings for Stateflow <p>The rationale from hisf_0010: Usage of transition paths (looping out of parent of source and destination objects) was incorporated into this guideline.</p>
hisf_0002: User-specified state/transition execution order	<p>New Model Advisor checks:</p> <ul style="list-style-type: none"> • DO-178C/DO-331: Check usage of Stateflow constructs • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check Stateflow charts for ordering of states and transitions • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check usage of Stateflow constructs
hisf_0009: Strong data typing (Simulink and Stateflow boundary)	<p>New Model Advisor check for DO-178C/DO-331: Check usage of Stateflow constructs</p>

High-Integrity Modeling Guideline	Description of Change
hisf_0011: Stateflow debugging settings	<p>New Model Advisor checks:</p> <ul style="list-style-type: none"> • DO-178C/DO-331: Check usage of Stateflow constructs • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check Stateflow debugging options • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check usage of Stateflow constructs
hisf_0013: Usage of transition paths (crossing parallel state boundaries)	<p>New Model Advisor checks:</p> <ul style="list-style-type: none"> • DO-178C/DO-331: Check Stateflow charts for transition paths that cross parallel state boundaries • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check Stateflow charts for transition paths that cross parallel state boundaries
hisf_0015: Strong data typing (casting variables and parameters in expressions)	<p>New Model Advisor checks:</p> <ul style="list-style-type: none"> • DO-178C/DO-331: Check Stateflow charts for strong data typing • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check Stateflow charts for strong data typing
hisf_0064: Shift operations for Stateflow data to improve code compliance	<p>New Model Advisor checks:</p> <ul style="list-style-type: none"> • DO-178C/DO-331: Check usage of shift operations for Stateflow data • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check usage of shift operations for Stateflow data <p>Title update. No change to guideline content.</p>

High-Integrity Modeling Guideline	Description of Change
hisf_0065: Type cast operations in Stateflow to improve code compliance	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check assignment operations in Stateflow charts • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check assignment operations in Stateflow charts Title update. No change to guideline content.
hisf_0211: Protect against use of unary operators in Stateflow Charts to improve code compliance	New Model Advisor checks: <ul style="list-style-type: none"> • DO-178C/DO-331: Check Stateflow charts for unary operators • IEC 61508, IEC 62304, EN 50128, and ISO 26262: Check Stateflow charts for unary operators Title update. No change to guideline content.

Modeling Support for Secure Coding Standards: Check model for compliance with secure coding requirements in CERT C, CWE, ISO/IEC TS 17961 standards to improve security of generated code

You can use Model Advisor to check the model or subsystem for compliance with secure coding requirements in CERT C, CWE, and ISO/IEC TS 17961 standards. To execute these checks, Select and Run Model Advisor Checks (Simulink) and select **By Task > Modeling Guidelines for Secure Coding (CERT C, CWE, ISO/IEC TS 17961)**.

This table summarizes the Modeling Standards for Secure Coding checks.

Check	Description	Addresses Secure Coding Standards
Check configuration parameters for secure coding standards	Identifies configuration parameters that might impact code security.	

Check	Description	Addresses Secure Coding Standards
Check for blocks not recommended for C/C++ production code deployment	Identifies blocks not supported by code generation or not recommended for C/C++ production code deployment.	
Check for blocks not recommended for secure coding standards	Identifies blocks not supported by secure coding standards.	
Check usage of Assignment blocks	Identifies Assignment blocks that do not have block parameter Action if any output element is not assigned set to Error or Warning	<ul style="list-style-type: none"> • ISO/IEC TS 17961: 2013, uninitref • CERT C, EXP33-C • CWE, CWE-908
Check for switch case expressions without a default case	Identifies switch case expressions that do not have a default case.	<ul style="list-style-type: none"> • ISO/IEC TS 17961: 2013, swtchdflt • CERT C, MSC01-C • CWE, CWE-478
Check for bitwise operations on signed integers	Identifies Simulink blocks that contain bitwise operations on signed integers. The check does not flag MATLAB Function or Stateflow blocks that use signed operands for bitwise operators.	<ul style="list-style-type: none"> • CERT C, INT13-C • CWE, CWE-682
Check for equality and inequality operations on floating-point values	Identifies equality and inequality operations on floating-point values.	<ul style="list-style-type: none"> • CERT C, FLP00-C • CWE, CWE-697
Check integer word length	Identifies integer word lengths that do not comply with hardware implementation settings.	<ul style="list-style-type: none"> • CERT C, INT13-C • CWE, CWE-682

If you have Simulink Design Verifier, the following design error detection checks are also available as part of the Modeling Standards for Secure Coding checks.

Check	Description	Addresses Secure Coding Standards
Detect Dead Logic	Identifies logic that stays inactive during simulation.	<ul style="list-style-type: none"> • CERT C, MSC07-C • CWE, CWE-561
Detect Integer Overflow	Identifies operations that exceed the data type range for integer or fixed-point operations.	<ul style="list-style-type: none"> • ISO/IEC TS 17961: 2013, intoflow • CERT C, INT30-C and INT32-C • CWE, CWE-190
Detect Division by Zero	Identifies operations in the model that cause division-by-zero errors.	<ul style="list-style-type: none"> • ISO/IEC TS 17961: 2013, diverr • CERT C, INT33-C and FLP03-C • CWE, CWE-369
Detect Out Of Bound Array Access	Detects operations that access outside the bounds of an array index	<ul style="list-style-type: none"> • ISO/IEC TS 17961: 2013, invptr • CERT C, ARR30-C • CWE, CWE-118
Detect Violation of Specified Minimum and Maximum Values	Checks the specified minimum and maximum values (the design ranges) on intermediate signals throughout the model and on the output ports. If the analysis detects that a signal exceeds the design range, the results identify where in the model the errors occurred.	<ul style="list-style-type: none"> • CERT C, API00-C • CWE, CWE-628

For information about the secure coding standards organizations, see Secure Coding Standards (Embedded Coder).

MISRA C: 2012 Modeling Checks: Improve compliance of generated code by using new MISRA C:2012 standards checks

To improve MISRA C:2012 compliance, these new checks are available through the Model Advisor. To execute these checks, Select and Run Model Advisor Checks (Simulink) and select **By Task > Modeling Guidelines for MISRA C:2012**.

Check	Description	Addresses MISRA C:2012
Check for missing error ports for AUTOSAR receiver interfaces	Identifies AUTOSAR receiver interface inports that do not have matching error ports.	Directive 4.7
Check for missing const qualifiers in model functions	Identifies input data pointers that do not have a const qualifier.	Rule 8.13
Check integer word length	Identifies integer word lengths that do not comply with hardware implementation settings.	Rule 10.1

Modifications to existing MISRA C:2012 compliance checks are outlined in this table.

Check	Description of Modification to the Check
Check for blocks not recommended for MISRA C:2012	Flags the inclusion of From Workspace blocks

Check	Description of Modification to the Check
Check configuration parameters for MISRA C:2012	Flags the following parameter settings: <ul style="list-style-type: none"> • Configuration parameter Wrap on overflow is set to none. • Configuration parameter Inf or NaN block output is set to none • Configuration parameter Dynamic memory allocation in MATLAB Function blocks is selected. • Parameter <code>ERTFilePackagingFormat</code> is set to <code>Modular</code>. • Parameter <code>PreserveStaticInFcnDecls</code> is set to <code>off</code>. <p>hisl_0060: Configuration parameters that improve MISRA C:2012 compliance reflects these parameter settings.</p>
Check for switch case expressions without a default case	Check can be executed on library models. Check can exclude blocks when you have Simulink Check.
Check for bitwise operations on signed integers	Check can exclude blocks when you have Simulink Check.
Check for equality and inequality operations on floating-point values	Check can exclude blocks when you have Simulink Check.

For information about MISRA C versions and updates, see MISRA C Guidelines (Embedded Coder).

DO-178C/DO-331 Modeling Checks: Removed Model Advisor check "Check model for block upgrade issues"

In R2017b, Model Advisor check Check model for block upgrade issues (check ID `mathworks.design.Update`) is no longer available under **Analysis > Model Advisor > Modeling Standards for DO-178C/DO-331 > Simulink**.

You can still execute this check through the Upgrade Advisor (Simulink) at **Analysis > Model Advisor > Upgrade Advisor**.

Model Metrics: Evaluate model quality by using new metric algorithms

Evaluate model quality by using these new model metrics:

- Simulink diagnostic warning count: Measures the number of Simulink diagnostic warnings reported during model compilation for simulation.
- Parameter count: Measures the number of parameters in a model.
- Simulink clone count: Measures the number of clones in a model.
- Clone component content: Quantifies cloned content in the model.
- Library linked component content: Quantifies library-linked content in the model.
- Stateflow chart count: Measures the number of Stateflow charts at the model level.
- MatlabFunction count: Measures the number of MATLAB Function blocks at the model level.
- Explicit IO count: Measures the number of inports and outports to and from the model.
- File Count: Measures the number of model and library files.
- Model file count: Measures the number of model files.

For more information on these, and other available metric algorithms, see Model Metrics.

Model Metric APIs: Create custom metrics with more detailed results and determine passed or failed compliance checks

In R2017b, the `slmetric.metric.Result` class contains the new property `Details`. `Details` is an array of objects of the new class `slmetric.metric.ResultDetail`. You can write custom metrics that use this new class to store details about what the `Value` property of the `slmetric.metric.Result` object counts. You can also use this class to determine which MAAB and DO-178C/DO-331 metrics passed or failed.

For existing classes, there are these new properties:

Class	New Property
<code>slmetric.metric.ResultDetail</code>	<code>Details</code>
<code>slmetric.metric.Result</code>	<code>ID</code>
<code>slmetric.metric.ResultCollection</code>	<code>Outdated</code>
<code>slmetric.metric.Metric</code>	<code>SupportsResultDetails</code>

For more information see, `slmetric.metric.ResultDetail`.

Compatibility Considerations

In R2017b, you cannot collect metric data for MISRA C:2012 and ISO 26262 metrics. Specifically, these metrics are not available:

- `mathworks.metrics.ModelAdvisorCheckCompliance.misra_c`
- `mathworks.metrics.ModelAdvisorCheckCompliance.ISO26262`
- `mathworks.metrics.ModelAdvisorCheckIssues.misra_c`
- `mathworks.metrics.ModelAdvisorCheckIssues.ISO26262`

For the DO-178C/DO-331 compliance metrics, the metric IDs `mathworks.metrics.ModelAdvisorCheckCompliance.do178` and `mathworks.metrics.ModelAdvisorCheckIssues.do178` are now named `mathworks.metrics.ModelAdvisorCheckCompliance.hisl_do178` and `mathworks.metrics.ModelAdvisorCheckIssues.hisl_do178`.

Model Advisor Configuration Editor: Select edit-time checks from folders

In the Model Advisor Configuration Editor, the tool now lists edit-time checks in folders instead of in a flat list. The folder structure is the same folder structure as for the Model Advisor. The Model Advisor Configuration Editor includes only folders that contain edit-time checks.

For more information, see [Organize Checks and Folders Using the Model Advisor Configuration Editor](#).

Model Metric APIs: Removed Model block architectural component

In R2018a, for a specified metric engine object, you can no longer collect metric data for Model blocks. However, you can still collect metric data for these Simulink objects:

- Model
- Subsystem block
- Chart
- MATLAB Function block
- Protected model

For custom metrics, in your `algorithm` method, you can no longer specify a `ComponentScope` that is a Model block.

Eliminating the Model block component does not mean that you are missing valuable data. The parent model `AggregatedValue` includes the data for the Model block `AggregatedValue`. The model block `Value` did not contain data.

For example, in R2017b, for the `sldemo_mdhref_basic` model, these are the results for the `mathworks.metrics.SimulinkBlockCount`.

```
ComponentPath: sldemo_mdhref_basic
  Value: 12
  AggregatedValue: 66
ComponentPath: sldemo_mdhref_basic/CounterA
  Value: NaN
  AggregatedValue: 18
ComponentPath: sldemo_mdhref_basic/CounterB
  Value: NaN
  AggregatedValue: 18
ComponentPath: sldemo_mdhref_basic/CounterC
  Value: NaN
  AggregatedValue: 18
ComponentPath: sldemo_mdhref_basic/More Info
  Value: 0
  AggregatedValue: 0
ComponentPath: sldemo_mdhref_counter
  Value: 18
  AggregatedValue: 18
```

The three instances of the referenced model `sldemo_mdhref_counter` (that is Counter A, Counter B, and Counter C) have results. They have a `Value` of `NaN` and the `sldemo_mdhref_basic` results include their aggregated values.

In R2018a, for the `sldemo_mdhref_basic` model, these are the results for the `mathworks.metrics.SimulinkBlockCount` metric:

```
ComponentPath: sldemo_mdhref_basic
  Value: 12
  AggregatedValue: 66
ComponentPath: sldemo_mdhref_basic/More Info
  Value: 0
  AggregatedValue: 0
ComponentPath: sldemo_mdhref_counter
  Value: 18
  AggregatedValue: 18
```

The results do not contain the individual instances of `sldemo_mdhref_counter`. The aggregated value of `sldemo_mdhref_basic` results still includes their aggregated values.