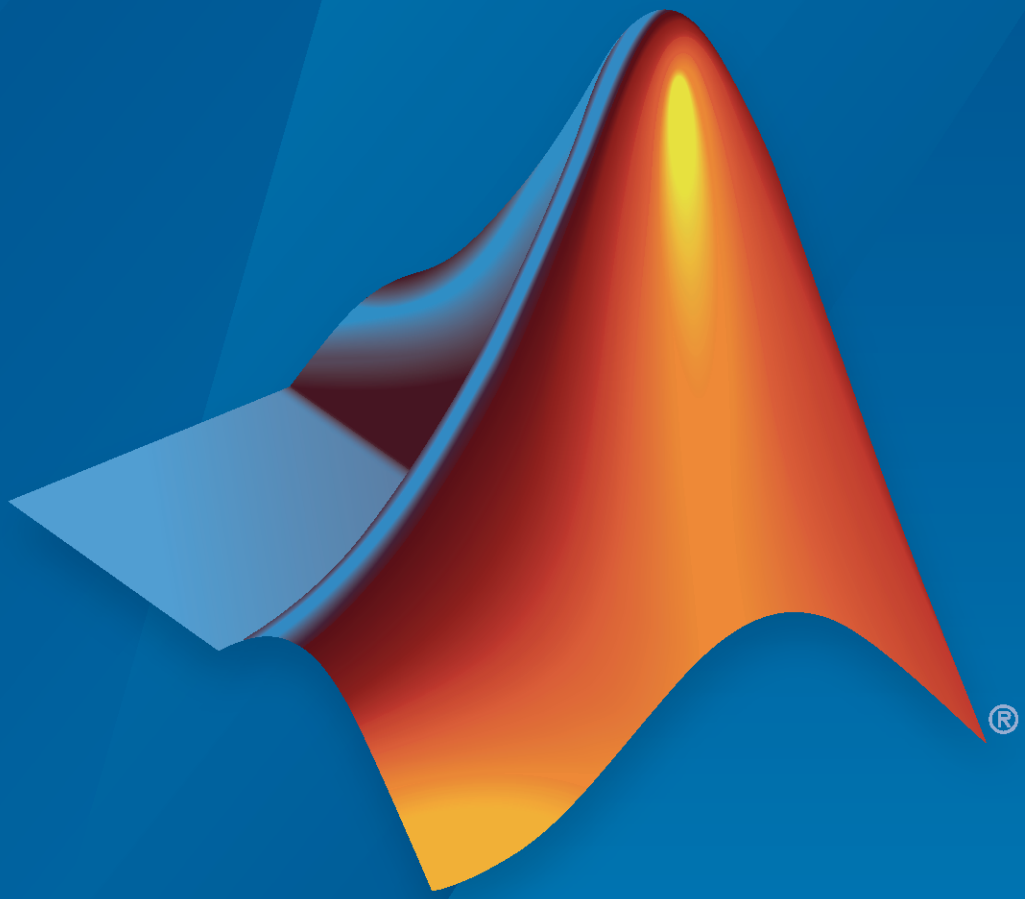
AUTOSAR Blockset Release Notes



MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

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

Phone: 508-647-7000

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

AUTOSAR Blockset Release Notes

Contents

R2020a

R2019b

R2019a

R2020a

AUTOSAR software component modeling

To improve AUTOSAR classic component modeling, R2020a adds:

- Basic Software blocks for modeling function inhibition
- Export of multidimensional matrices for AUTOSAR variables

Model function inhibition by using Basic Software blocks

For the AUTOSAR Classic Platform, AUTOSAR Blockset provides Basic Software (BSW) blocks, which allow you to model software component calls to BSW services that run in the AUTOSAR run-time environment. BSW services include NVRAM Manager and Diagnostic Event Manager.

R2020a extends BSW service support to include component calls to Function Inhibition Manager (FiM) services. As defined in the AUTOSAR specification, the Function Inhibition Manager provides a control mechanism for selectively inhibiting (that is, deactivating) function execution in software component runnables, based on function identifiers (FIDs) with inhibition conditions. For example, an FID can represent functionality that must be stopped if a specific failure occurs.

The Function Inhibition Manager is closely related to the Diagnostic Event Manager (Dem), because inhibition conditions can be based on the status of diagnostic events. For example, if a sensor failure event is reported to the Diagnostic Event Manager, the Function Inhibition Manager can inhibit the associated function identifier and stop execution of the corresponding functionality.

R2020a adds FiM and Dem blocks that allow you to:

- Query the status of function inhibition conditions.
- Configure function inhibition criteria based on diagnostic event status.
- Define operation cycles to scope failures to a time period.

For more information, see “Configure Calls to AUTOSAR Function Inhibition Manager Service”.

Export multidimensional matrices for AUTOSAR variables

For an AUTOSAR component model with multidimensional arrays, if you set the model configuration parameter **Array layout** to `Row-major`, you can preserve dimensions of multidimensional arrays in the generated C code. Preserving array dimensions in the generated code can enhance code integration.

R2020a improves ARXML export to honor the row-major setting for multidimensional signals and states mapped to AUTOSAR variables. If **Array layout** is set to `Row-major`, ARXML export no longer flattens multidimensional matrices in the generated descriptions for signals and states mapped to AUTOSAR variables.

In the generated C code, access to the variable data depends on the type of storage. For internal storage, such as static memory, the code uses multidimensional indexing. For external storage, the code calls AUTOSAR `Rte` functions, which use one-dimensional indexing.

When you use `Row-major` array layout, you can disregard the setting of the AUTOSAR model configuration parameter **Support root-level matrix I/O using one-dimensional arrays**. **Support root-level matrix I/O using one-dimensional arrays** provides a workaround for exporting multidimensional matrices in `Column-major` array layout.

AUTOSAR adaptive software component modeling

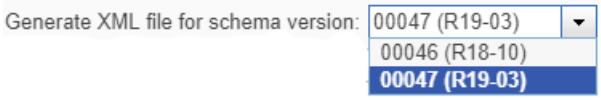
To improve AUTOSAR adaptive component modeling, R2020a adds:

- Adaptive Platform Release 19-03 support
- Data calibration based on XCP communication and ASAP2 file generation
- Dynamic service discovery

Support for AUTOSAR Adaptive Platform Release 19-03

R2020a extends support of the AUTOSAR Adaptive Platform to include Release 19-03. AUTOSAR Blockset supports the 000047 (R19-03) schema for import and export of ARXML files and generation of AUTOSAR-compliant C++ code.

- In R2020a, 000047 (R19-03) is the default schema version for AUTOSAR adaptive models created in Simulink®.



- If you import schema 000047 (R19-03) ARXML files into Simulink, the ARXML importer detects and uses the schema version and sets the schema version parameter in the model.
- Building an AUTOSAR adaptive model using schema 000047 (R19-03) generates ARXML descriptions and C++ code that comply with Adaptive Platform Release 19-03.

For more information on AUTOSAR schema versions, see “Select AUTOSAR Schema”.

Calibrate data for adaptive applications by using XCP and ASAP2

R2020a allows you to configure run-time calibration of adaptive application data based on XCP Slave communication and ASAP2 (A2L) file generation. The XCP and ASAP2 capabilities are defined outside the Adaptive Platform (AP) specifications, which as of Release 19-03 do not address data calibration.

As part of generating and deploying adaptive code, you can configure interfaces for XCP Slave communication in the generated C++ code and export A2L files containing model data for measurement and calibration.

Before deploying adaptive code, you can:

- Use the Configuration Parameters dialog box to configure the model, to generate XCP Slave function calls in adaptive C++ code.
- Use the ASAP2 Generator app to configure and generate an ASAP2 (A2L) file that describes model data for measurement and calibration.

For more information, see “Configure AUTOSAR Adaptive Data for Run-Time Measurement and Calibration”.

Find adaptive services by using dynamic discovery

R2020a allows you to configure applications to use dynamic discovery to subscribe to adaptive services as they become available. Previously, applications found and subscribed to adaptive services one time during initialization. One-time discovery may require adaptive services to start before applications and prevent applications from using new services as they become available. Now you can

configure, in your model or programmatically, the service discovery mode of each required service port as `OneTime` or `DynamicDiscovery`.

For more information, see “Configure AUTOSAR Adaptive Service Discovery Modes”.

AUTOSAR software architecture modeling

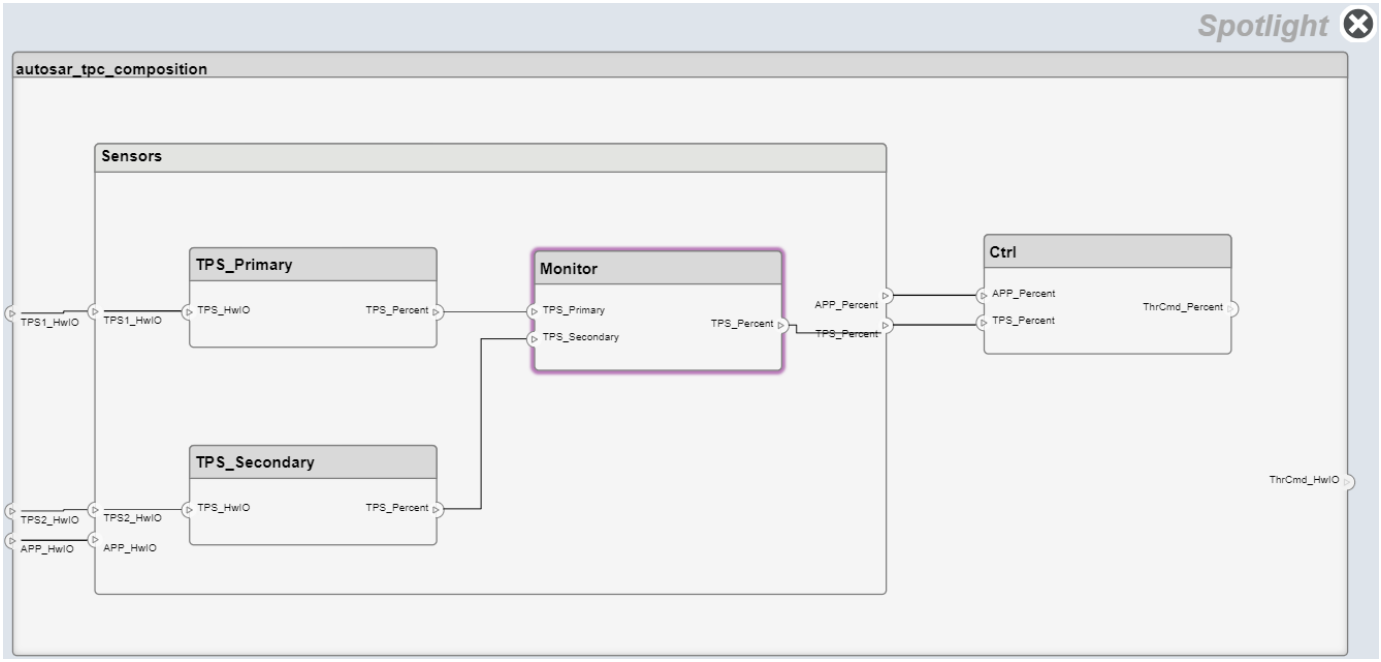
To improve AUTOSAR software architecture modeling, R2020a adds:

- Spotlight views for analyzing dependencies
- Programmatic interface for authoring architecture models

Use spotlight view to analyze component or composition dependencies

In an AUTOSAR architecture model, to help analyze component or composition dependencies, you can create a spotlight view. A spotlight view is a simplified view of an architecture component or composition that captures its upstream and downstream dependencies.

Here is a spotlight view of component `Monitor` in AUTOSAR example model `autosar_tpc_composition`.



For more information, see “View AUTOSAR Component or Composition Dependencies”.

Programmatically create and configure architecture models

R2020a adds a programmatic interface for developing AUTOSAR architecture models. Use the AUTOSAR architecture functions to:

- Create, load, open, save, or close an AUTOSAR architecture model.
- Add, connect, or remove AUTOSAR components, composition, and ports.

- Find AUTOSAR elements and modify properties.
- Define component behavior by creating or linking Simulink models.
- Add Basic Software (BSW) service component blocks for simulating BSW service calls.
- Export composition and component ARXML descriptions and generate component code (requires Embedded Coder®).

For more information, see “Software Architecture Modeling” and “Configure AUTOSAR Architecture Model Programmatically”.

Functionality being removed or changed

Support for AUTOSAR Classic Platform schemas 3.x and 2.1 will be removed
Still runs

Support for AUTOSAR Classic Platform schemas 3.x and 2.1 will be removed in a future release. Use schema 4.0 or later instead. The default schema for new AUTOSAR models is 4.3. For more information, see “Select AUTOSAR Schema”.

R2019b

Version: 2.1

New Features

Bug Fixes

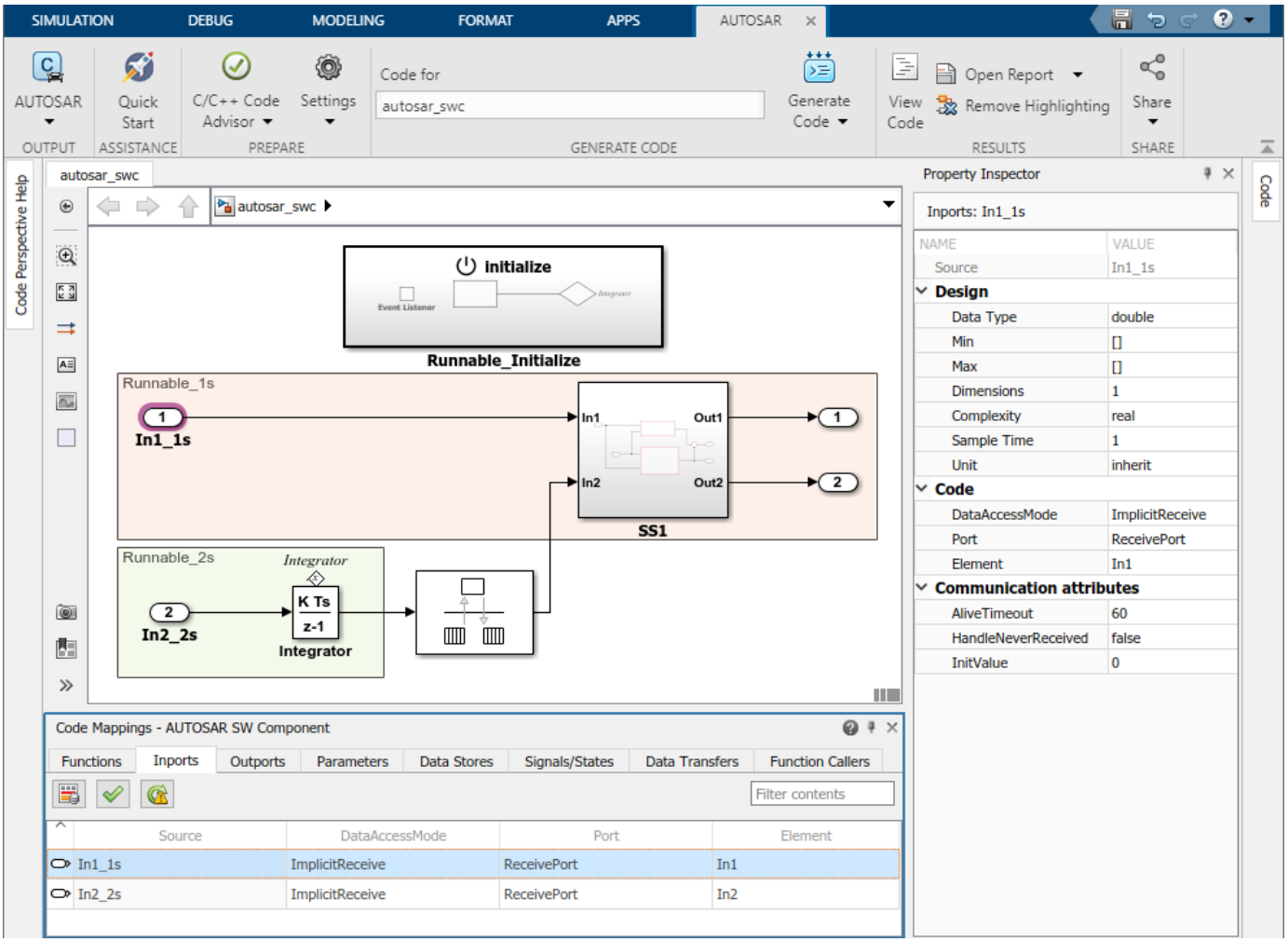
AUTOSAR Component Designer app and AUTOSAR tab

To support AUTOSAR software component modeling, R2019b introduces an AUTOSAR Component Designer app and an **AUTOSAR** tab. The app and the tab support common tasks for component-level AUTOSAR software development.

The AUTOSAR Component Designer app opens an AUTOSAR code perspective, which displays the **AUTOSAR** tab, a help panel, a Property Inspector dialog box, and, directly under the model, the Code Mappings editor.

To use the AUTOSAR Component Designer app, open a component model. On the **Apps** tab, click **AUTOSAR Component Designer**.

- If the model has a mapped AUTOSAR software component, the app opens the AUTOSAR code perspective, with the **AUTOSAR** tab displayed.
- If the model does not have a mapped AUTOSAR software component, the app opens the AUTOSAR Component Quick Start. Use the AUTOSAR Component Quick Start to configure the model for the AUTOSAR Classic or Adaptive Platform. When you complete the quick-start procedure and click **Finish**, your model opens in the AUTOSAR code perspective, with the **AUTOSAR** tab displayed.



AUTOSAR software component modeling

To improve AUTOSAR classic component modeling, R2019b adds submodel data mapping, variation points for calibration data, intuitive port modeling with Simulink bus ports, and runnable scheduling with the Schedule Editor.

Map calibration data for submodels referenced from component models

In R2019b, the Code Mappings editor adds support for submodels referenced from AUTOSAR software component models. In an open submodel, you can:

- Configure the submodel as a model referenced from an AUTOSAR software component model. Use the AUTOSAR Component Quick Start or the AUTOSAR function `autosar.api.create`.
- In the AUTOSAR code perspective, use the Code Mappings editor to configure the submodel internal data.
- To generate C code and AUTOSAR XML (ARXML) files that support run-time calibration of the submodel internal data, open and build the component model that references the submodel.

For more information, see Map Calibration Data for Submodels Referenced from AUTOSAR Component Models.

Export variation points for calibration data

In R2019b, you can export variation points for AUTOSAR calibration data, including:

- Parameters — Calibration, shared internal, instance-specific, or constant memory
- Per-instance memory — C- typed or AR-typed
- Inter-runnable variables (IRVs) — Implicit or explicit

You can model calibration data in combination with different types of variant conditions. Model the variant conditions by using Variant Source and Variant Sink blocks, Variant Subsystem blocks, or model reference variants. When you build your model, the exported AUTOSAR XML (ARXML) files contain the conditionally used data elements and their variation points.

For more information, see Export Variation Points for AUTOSAR Calibration Data.

Model AUTOSAR ports by using Simulink bus ports

In R2019b, you can model AUTOSAR ports by using Simulink bus ports instead of Simulink signal ports. Bus port blocks In Bus Element and Out Bus Element can simplify model interfaces. For more information, see Simplify Bus Interfaces (Simulink).

Bus ports provide a more intuitive way to model AUTOSAR communication ports, interfaces, and groups of data elements. If you model AUTOSAR ports with In Bus Element and Out Bus Element blocks, and type the bus ports with bus objects, basic properties of AUTOSAR ports, interfaces, and data elements are configured without using the AUTOSAR Dictionary. For more information, see Configure AUTOSAR Ports By Using Simulink Bus Ports.

In an AUTOSAR architecture model, if you link to an existing software component model that uses root Inport and Outport blocks, the software automatically converts the signal ports to bus ports.

Configure runnable execution order by using Schedule Editor

In R2019b, you can use the Schedule Editor to schedule and specify the execution order of AUTOSAR component runnables for simulation. In the Schedule Editor, you can:

- View a graphical representation of runnables as partitions in an AUTOSAR component.
- Directly specify the execution order of runnables.

For more information, see Using the Schedule Editor (Simulink) and Configure AUTOSAR Runnable Execution Order By Using Schedule Editor.

You can also use the Schedule Editor in AUTOSAR architecture modeling. For more information, see Configure AUTOSAR Scheduling and Simulation.

Code Mappings editor changes

These changes were made to the Code Mappings editor tabs:

- The **Entry-Point Functions** tab was renamed to **Functions**.
- The tab order, from left to right, was changed to **Functions**, **Inports**, **Outports**, **Parameters**, **Data Stores**, **Signals/States**, **Data Transfers**, and **Function Callers**.
- On the **Parameters** tab, parameter categories were renamed.
  - **Local parameters** was renamed to **Model parameters**.
  - **Parameter arguments** was renamed to **Model parameter arguments**.

Also, the Code Mappings editor now retains mapping information when you perform cut/copy/paste or undo/redo operations on data stores, states, or signals in the model diagram. You can:

- Cut/copy and paste a data store, state, or signal. Code Mapping Editor copies the **Mapped To** information for the element. A signal cut or copy must include the signal line and the originating block.
- Undo/redo an operation on a data store, state, or signal. **Mapped To** information is retained.

AUTOSAR adaptive software component modeling

To improve AUTOSAR adaptive component modeling, R2019b adds ARXML import, service instance identification, and memory allocation for event sends.

Import ARXML software descriptions

In R2019b, you can import AUTOSAR XML (ARXML) descriptions of adaptive software components, service interfaces, and data types into Simulink. Use the ARXML importer to:

- Create an initial Simulink representation of an AUTOSAR adaptive software component.
- Update a mapped AUTOSAR adaptive component model with shared ARXML definitions of service interfaces and data types.

You can participate in round-trip exchanges of adaptive component ARXML descriptions between Simulink and other development environments.

For more information, see Import AUTOSAR Adaptive Software Descriptions.

Configure service instance identification for ARXML manifest and generated code

In R2019b, you can configure service instance identification for AUTOSAR required and provided ports. When you build an adaptive software component model:

- Exported ARXML files include a service instance manifest file, which describes port-to-service instance mapping.
- Generated C++ code uses the configured service instance information in `ara::com` function calls.

For more information, see Configure AUTOSAR Adaptive Service Instance Identification.

Configure event sends with memory allocation

To send service event data, the AUTOSAR Adaptive Platform supports these methods:

- By reference — The send function uses memory in the application address space. After the send returns, the application can modify the event data.
- By `ara::com` allocated memory — The application requests `ara::com` middleware to allocate memory for the data. This method avoids data copies by `ara::com` middleware and can be more efficient for frequent sends or large amounts of data. But the application loses access to the memory after the send returns.

In R2019b, you can configure adaptive event sends to request `ara::com` memory allocation. Previously, all event sends were by reference.

For more information, see Configure Memory Allocation for AUTOSAR Adaptive Service Data.

AUTOSAR software architecture modeling

R2019b introduces AUTOSAR software architecture modeling for the Classic Platform (requires System Composer™). Using a Simulink Start Page template, you create an architecture model. Within the architecture model, you add compositions and components and link new or existing models. You simulate the behavior of the aggregated components. If you have Embedded Coder software, you can export composition and component AUTOSAR XML (ARXML) descriptions and generate component code.

Create architecture models

In R2019b, you can create an AUTOSAR architecture model, which provides resources and a canvas for developing AUTOSAR composition and component models for the Classic Platform. Without leaving the architecture model, you can:

- Add and connect AUTOSAR compositions and components.
- Link components to requirements (requires Simulink Requirements™).
- Define component behavior by creating or linking Simulink models.
- Configure scheduling and simulation.
- Generate and package composition ARXML descriptions and component code (requires Embedded Coder)

Architecture models provide an end-to-end, top-to-bottom AUTOSAR software design workflow. In Simulink, you can author a high-level application design, implement behavior for application components, add Basic Software (BSW) service calls and service implementations, and simulate the application.

For more information, see Create AUTOSAR Architecture Models.

Add and connect compositions and components

After you create an AUTOSAR architecture model, use the composition editor and the Simulink Toolstrip **Modeling** tab to add and connect compositions and components. Common tasks include:

- Add Component and Composition blocks from the palette or toolstrip.
- Create ports by clicking Component or Composition block edges.
- Connect Component and Composition blocks by dragging signal lines.
- Configure additional AUTOSAR properties by using the Property Inspector.

For more information, see Add and Connect AUTOSAR Compositions and Components.

Define component behavior by creating or linking models

After you add and connect Component and Composition blocks in an AUTOSAR architecture model, you can add Simulink behavior to the components. For each AUTOSAR Component block, you can:

- Create a model based on the block interface.
- Link to an implementation model.
- Create a model from an AUTOSAR XML (ARXML) component description.

For more information, see Define AUTOSAR Component Behavior by Creating or Linking Models.

Configure scheduling and simulation

To simulate the behavior of the aggregated components in an AUTOSAR architecture model, click **Run**. To configure scheduling and simulation, you can:

- Add Basic Software (BSW) blocks, including Diagnostic Service Component and NVRAM Service Component blocks, to simulate calls to BSW services.
- Create a test harness model to connect inputs and plant elements to the architecture model.
- Use the Schedule Editor to schedule and specify the execution order of component runnables for simulation.

For more information, see Configure AUTOSAR Scheduling and Simulation.

Generate and package composition ARXML descriptions and component code

In AUTOSAR architecture models, with one click, you can export composition and component ARXML descriptions, generate component code, and package build artifacts. If you initiate an export that encompasses a composition, it generates:

- XML descriptions for compositions, component prototypes, ports, and connectors.
- AUTOSAR compliant code for components.
- A zip file that packages ARXML files, code files, and required artifacts for integration with an AUTOSAR run-time environment.

For more information, see Generate and Package AUTOSAR Composition XML Descriptions and Component Code.

R2019a

Introducing AUTOSAR Blockset

In R2019a, the AUTOSAR Blockset product replaces the Embedded Coder Support Package for AUTOSAR Standard. You use AUTOSAR Blockset to design and simulate AUTOSAR software.

AUTOSAR Blockset provides an AUTOSAR dictionary and blocks for developing Classic and Adaptive AUTOSAR software using Simulink models. You can define AUTOSAR software component properties, interfaces, and data types, and map them to existing Simulink models using the AUTOSAR editor. Alternatively, the blockset provides an application interface that lets you automatically generate new Simulink models for AUTOSAR by importing software component and composition descriptions from AUTOSAR XML files.

AUTOSAR Blockset provides blocks and constructs for AUTOSAR library routines and Basic Software (BSW) services, including Memory Access and Diagnostics. By simulating the BSW services together with your application software model, you can verify your AUTOSAR ECU software without leaving Simulink.

AUTOSAR Blockset supports C and C++ production code generation and AUTOSAR XML file export (with Embedded Coder). The software is qualified for use with the ISO 26262 standard (with IEC Certification Kit).

Product restructuring overview

In R2019a, AUTOSAR Blockset provides AUTOSAR software design and simulation support that previously was provided by Embedded Coder and the Embedded Coder Support Package for AUTOSAR Standard. In the new product, in general, software interfaces and development workflows are unchanged from previous releases. Product restructuring introduced these differences:

- Product requirements and dependencies:
  - AUTOSAR Blockset requires only MATLAB® and Simulink.
  - Embedded Coder is required to generate AUTOSAR C/C++ code and XML files.
  - No support package is required.
- Second development platform:
  - R2019a adds AUTOSAR Adaptive Platform support to existing Classic Platform support.
  - Classic and adaptive development workflows are logically distinct, and are handled separately in AUTOSAR Blockset software interfaces and documentation.
  - Existing Classic Platform workflows are unchanged, except where enhanced by R2019a features.
- AUTOSAR block libraries relocated:
  - AUTOSAR Blockset libraries replace block libraries from Embedded Coder and Embedded Coder Support Package for AUTOSAR Standard.
  - Existing block names are unchanged; new blocks are added.
- AUTOSAR example models renamed and relocated:
  - For example, models `autosar_swc*.slx` in folder `examples/autosarblockset` replace models `rtwdemo_autosar_swc*.slx`, formerly in folder `toolbox/rtw/rtwdemos`.
  - Models remain accessible from the MATLAB command line.

- AUTOSAR HTML help and PDF books restructured and relocated:
  - AUTOSAR Blockset HTML help replaces AUTOSAR help in Embedded Coder and Embedded Coder Support Package for AUTOSAR Standard.
  - AUTOSAR Blockset Reference, User's Guide, and Release Notes PDF books replace the Embedded Coder AUTOSAR PDF book.

Resources for Upgrading from AUTOSAR Standard Support Package

If you are upgrading to AUTOSAR Blockset from Embedded Coder Support Package for AUTOSAR Standard, review information about compatibility and upgrade issues at the following locations:

- AUTOSAR Blockset Release Notes (latest release). In HTML notes, to view only compatibility considerations, select **Incompatibilities Only**.
- On the MathWorks® web site, view the R2018b Embedded Coder Support Package for AUTOSAR Standard Release Notes. To view only compatibility considerations, select **Incompatibilities Only**. This document provides compatibility information for releases up through R2018b.

You can also refer to the rest of the archived documentation, including release notes, for Embedded Coder Support Package for AUTOSAR Standard and Embedded Coder.

Compatibility Considerations

If you are upgrading to R2019a AUTOSAR Blockset from Embedded Coder Support Package for AUTOSAR Standard, these minor compatibility considerations might apply.

- R2019a changes AUTOSAR example model locations. If you reference model names in scripts, update the model names. The new example models use the prefix `autosar_` and are located in the folder *`matlabroot`*`/examples/autosarblockset`.
- R2019a restructures and relocates AUTOSAR HTML help and PDF books. Consider updating any bookmarks or references.

For function interface compatibility considerations related to R2019a new features, see “Incrementally auto-configure and map new Simulink elements in AUTOSAR model” on page 3-5.

For function interface changes that support R2019a workflow improvements, consider migrating to the improved workflows. For more information, see “Reuse existing AUTOSAR elements for software components created in Simulink” on page 3-5 and the lookup tables subsection of “Code Perspective enhancements for mapping data stores, model workspace parameters, and internal signals and states” on page 3-6.

AUTOSAR Classic Platform support extended to Release 4.3.1

R2019a extends support of AUTOSAR Classic Platform schema version 4.3 to include schema revision 4.3.1. AUTOSAR Blockset supports the new schema revision for import and export of ARXML files and generation of AUTOSAR-compatible C code.

If you import schema 4.3.1 ARXML code into Simulink, the ARXML importer detects and uses the schema version and revision, and sets the schema version parameter in the model. For more information on schema import and export, see Select an AUTOSAR Schema.

Support for AUTOSAR Adaptive Platform Release 18.10

In R2019a, you can flexibly model the structure and behavior of software components for the AUTOSAR Adaptive Platform. The Adaptive Platform defines a service-oriented architecture for automotive components that must flexibly adapt to external events and conditions. AUTOSAR Blockset supports Adaptive Platform Release 18.10.

You can:

- Model AUTOSAR adaptive software components using event-based communication.
- With Embedded Coder, generate C++ code compliant with the AUTOSAR Adaptive Platform.
- Deploy generated code and build a target executable.

For more information, see Model AUTOSAR Adaptive Software Components and Configure AUTOSAR Adaptive Software Components.

Generate AUTOSAR IFL and IFX library routines for interpolation using AUTOSAR lookup table blocks

R2019a provides AUTOSAR lookup table blocks that you can configure to generate specific IFL and IFX interpolation routines in your AUTOSAR-compliant C code.

- Curve – Approximate one-dimensional function
- Curve Using Prelookup – Use precalculated index and fraction values to accelerate approximation of one-dimensional function
- Map – Approximate two-dimensional function
- Map Using Prelookup – Use precalculated index and fraction values to accelerate approximation of two-dimensional function
- Prelookup – Compute index and fraction for Curve Using Prelookup or Map Using Prelookup block

To configure AUTOSAR library routine generation for a lookup table block, open the block parameters dialog box. Modify the block parameters to configure a specific AUTOSAR routine. If you select the AUTOSAR 4.0 code replacement library (CRL) for your model, code generated from the block is replaced with the AUTOSAR library routine that you configured. For more information, see Configure Lookup Tables for AUTOSAR Measurement and Calibration and Code Generation with AUTOSAR Code Replacement Library.

Enhanced AUTOSAR model creation in Simulink using Component Quick Start or Simulink Start Page

R2019a provides new resources for creating an AUTOSAR software component model in Simulink:

- AUTOSAR Component Quick Start – Provides easy, ordered steps for mapping an open Simulink model to an AUTOSAR software component. In an open model, set **System target file** to an AUTOSAR target and open Code perspective. AUTOSAR Component Quick Start opens, and you can quickly step through AUTOSAR software component configuration.
- Simulink Start Page – Provides AUTOSAR Blockset model templates. Select a Classic Platform or Adaptive Platform template, which you can use as a starting point for developing an AUTOSAR software component model.

For more information, see Create AUTOSAR Software Component in Simulink.

Reuse existing AUTOSAR elements for software components created in Simulink

R2019a enhances design and development of AUTOSAR software components created in Simulink by supporting generalized reuse of existing AUTOSAR element definitions. Changes include:

- Improved import of AUTOSAR element definitions with function `updateAUTOSARProperties`
- Option to treat imported elements as read-only (the default), preventing definition changes, or read-write

```
% Import XML definitions of AUTOSAR software address methods as read-write elements
ar = arxml.importer('SwAddressMethods.arxml');
updateAUTOSARProperties(ar,'mySWC','ReadOnly',false);
```

- AUTOSAR Dictionary displays **Exported XML File** name for packageable elements
- AUTOSAR XML export preserves ARXML file structure for imported noncomponent elements
- AUTOSAR property functions `createEnumeration` and `createNumericType` generate Simulink definitions for working with imported AUTOSAR elements
- AUTOSAR Component Quick Start supports import of AUTOSAR element definitions as part of initial configuration

Function `updateAUTOSARProperties` generalizes and replaces function `updateReferences`. If you previously used `updateReferences` to import existing AUTOSAR elements into a Simulink-originated component, consider replacing `updateReferences` function calls with `updateAUTOSARProperties` function calls.

For more information, see the `updateAUTOSARProperties` reference page, Reuse AUTOSAR Element Descriptions, and example Reuse AUTOSAR Elements in Component Model.

Incrementally auto-configure and map new Simulink elements in AUTOSAR model

R2019a enhances AUTOSAR function `autosar.api.create` so that you can incrementally configure and map Simulink elements as you modify your AUTOSAR model. When used with a mapped AUTOSAR model, function call `autosar.api.create(modelName)`:

- Preserves current model configuration and mapping.
- Finds and maps unmapped model elements.
- Updates the AUTOSAR Dictionary for deleted model elements.

For more information, see Incrementally Update AUTOSAR Mapping after Model Changes.

Compatibility Considerations

R2019a changes the default behavior of `autosar.api.create`. If you call the function without a *`mode`* argument (`init`, `default`, or `incremental`), the function behavior depends on the mapping state of the model.

- If the model is not mapped to an AUTOSAR software component, the function creates a Simulink to AUTOSAR mapping in `default` mode. In this mapping, Simulink inports and outports are mapped to AUTOSAR ports with default AUTOSAR properties.
- If the model is already mapped to an AUTOSAR software component, the function updates the existing mapping in `incremental` mode. The function finds and maps unmapped model elements, and updates the AUTOSAR Dictionary for deleted model elements.

Previously, if you called the function without a *`mode`* argument, the function created a Simulink to AUTOSAR mapping in `init` mode. In this mapping, Simulink inports and outports were not mapped to AUTOSAR ports. Also, if the model was already mapped to an AUTOSAR software component, the new mapping replaced the existing mapping.

If a MATLAB script previously called the function `autosar.api.create` without a *`mode`* argument, update the script to account for the R2019a behavior change. Consider taking advantage of the new behavior or, if you rely on the previous `init` mode behavior, specify `init` as the *`mode`* argument.

Code Perspective enhancements for mapping data stores, model workspace parameters, and internal signals and states

R2019a improves the Simulink to AUTOSAR mapping workflow by enhancing Code Mappings editor tabs and improving graphical display of model data.

Map data stores to AUTOSAR component per-instance and static memory for calibration

In R2019a, AUTOSAR Code Mappings editor adds a **Data Stores** tab. In AUTOSAR code perspective, use the tab to map internal data store elements to AUTOSAR component per-instance and static memory for calibration. Alternatively, you can use equivalent AUTOSAR map functions `getDataStore` and `mapDataStore`. For more information, see Map Data Stores to AUTOSAR Variables, Configure AUTOSAR Per-Instance Memory, and Configure AUTOSAR Static Memory.

Map model workspace parameters to AUTOSAR component instance-specific parameters for calibration

R2019a extends model workspace parameter mapping to support AUTOSAR per-instance parameters. Using Code Mappings editor, **Parameters** tab, you can map model workspace parameters that are marked as model arguments to AUTOSAR instance-specific parameters for calibration. Alternatively, you can use equivalent AUTOSAR map functions `getParameter` and `mapParameter`. For more information, see Map Model Workspace Parameters to AUTOSAR Component Internal Parameters, Shared and Per-Instance Parameters, and Configure AUTOSAR Shared or Per-Instance Parameters.

Signals and States tabs combined

R2019a combines AUTOSAR Code Mappings editor tabs **Signals** and **States** into a **Signals/States** tab. Use the **Signals/States** tab to map Simulink block signals or states to AUTOSAR variables for calibration.

AUTOSAR map functions `getSignal`, `getState`, `mapSignal`, and `mapState` are not affected by the **Signals/States** tab change.

Lookup Tables tab removed

R2019a removes the **Lookup Tables** tab from AUTOSAR Code Mappings editor. The **Lookup Tables** tab was used to map Simulink lookup table or breakpoint objects created in the base workspace to

AUTOSAR calibration parameters for integrated and distributed lookups. Instead, you can use the **Parameters** tab to map lookup table or breakpoint objects created in the model workspace. See Map Model Workspace Parameters to AUTOSAR Component Internal Parameters.

AUTOSAR map functions `getLookupTable` and `mapLookupTable` are not affected by the **Lookup Tables** tab removal. However, consider switching from using base workspace objects with functions `getLookupTable` and `mapLookupTable` to using model workspace objects with functions `getParameter` and `mapParameter`.

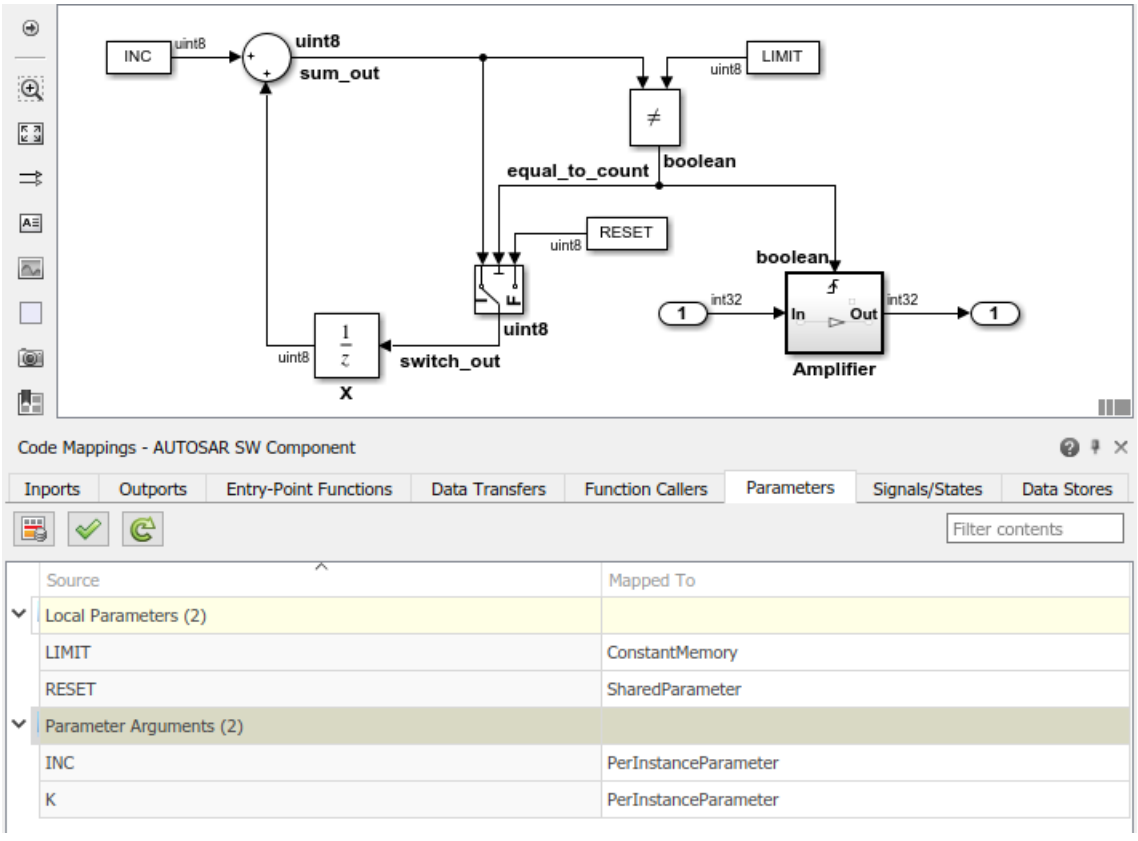
For an example of the model workspace-based workflow, see Configure Lookup Tables for AUTOSAR Measurement and Calibration.

Model data grouped in categories for easy reference

To improve navigation and usability of model data mapping tables, Code Mappings editor now groups model data into categories. The editor displays a node for each category. For example:

- The **Signals/States** tab displays nodes for **Signals** and **States**.
- The **Parameters** tab displays nodes for **Local Parameters** and **Parameter Arguments**.

To focus on entries of interest and reduce scrolling, you can expand and collapse model data categories.



Change to AUTOSAR XML import behavior for ArTypedPerInstanceMemory element with Service Dependency

In R2019a, when you import an ArTypedPerInstanceMemory element with a Service Dependency from an ARXML file into Simulink, the importer:

- Adds a Data Store Memory block to the model.
- Adds a corresponding `Simulink.Signal` object to the model workspace.
- Maps the internal data store to AUTOSAR component per-instance memory for calibration.

Previously, the importer added a Data Store Memory block to the model and added a corresponding `AUTOSAR.Signal` object to the base workspace.

Compatibility Considerations

If existing AUTOSAR infrastructure expects import of an ArTypedPerInstanceMemory element with a Service Dependency to add an `AUTOSAR.Signal` object to the base workspace, update the infrastructure to reflect the new importer behavior.

Model Advisor checks for AUTOSAR Blockset configuration and lookup table block code replacements

The following table identifies the new Model Advisor checks for AUTOSAR Blockset:

Model Advisor Check	Check ID
Check model configuration parameters for AUTOSAR compliance	mathworks.autosar.autosar_configset
Check compatibility of AUTOSAR Interpolation Routines	mathworks.autosar.lut_replacement_check

AUTOSAR contextual tab in the Simulink Toolstrip Tech Preview

In R2019a, you have the option to turn on the Simulink Toolstrip. For more information, see Simulink Toolstrip Tech Preview replaces menus and toolbars in the Simulink Desktop.

The Simulink Toolstrip includes contextual tabs, which appear only when you need them. The AUTOSAR contextual tab includes options for completing actions that apply only to AUTOSAR Blockset.

To access the AUTOSAR tab, with the Simulink Toolstrip activated, click **Apps** and then click the **AUTOSAR** button. The **AUTOSAR** tab appears.

Documentation does not reflect the addition of the AUTOSAR contextual tab.