

System Composer™

Reference



MATLAB® & SIMULINK®

R2019a



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

System Composer™ Reference

© COPYRIGHT 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.

Revision History

March 2019 Online only New for Version 1.0 (Release 2019a)

1 | Functions – Alphabetical List

2 | Classes – Alphabetical List

Functions — Alphabetical List

addChoice

Add a variant choice to a variant component

Syntax

```
compList = addChoice(variantComponent,choices)
compList = addChoice(variantComponent,choices,labels)
```

Description

`compList = addChoice(variantComponent,choices)` creates variant choices specified in `choices` in the specified variant component and returns their handles.

`compList = addChoice(variantComponent,choices,labels)` creates variant choices specified in `choices` with labels `labels` in the specified variant component and returns their handles.

Input Arguments

variantComponent — Architecture component
component

The architecture where the variant choices are added.

Data Types: `systemcomposer.arch.Component`

choices — Variant choice names
cell array of strings

Cell array where each element defines the name of a choice component. The length of `choices` must be the same as `labels`.

Data Types: `string`

labels — Variant choice labels
cell array of strings

Array of labels where each element is the label for the corresponding choice.. The length of labels must be the same as choices.

Data Types: `string`

Output Arguments

compList — Created components

array of components

Array of created components. This array is the same size as choices and labels.

See Also

`getActiveChoice` | `getChoices` | `makeVariant`

Topics

“Create Variants”

Introduced in R2019a

addComponent

Add a component to the architecture

Syntax

```
components = addComponent(architecture, compNames)
components = addComponent(architecture, compNames, stereotypes)
```

Description

`components = addComponent(architecture, compNames)` adds a set of components specified by the array of names.

`components = addComponent(architecture, compNames, stereotypes)` applies stereotypes specified in the `stereotypes` to the new components.

Examples

Create a Model with two Components

Create model, get root architecture, and create components.

```
model = systemcomposer.createModel('archModel');
arch = get(model, 'Architecture');
names = {'Component1', 'Component2'}
comp = addComponent(arch, names);
```

Input Arguments

architecture — Architecture model element

`architecture`

Parent architecture to which the component is added.

Data Types: `systemcomposer.arch.Architecture`

compNames — Names of components

cell array of strings

Cell array where each element defines the name of a new component. The length of `compNames` must be the same as `stereotypes`.

Data Types: `string`

stereotypes — Stereotypes to apply to the components

cell array of stereotypes

Array of stereotypes where each element is the qualified stereotype name for the corresponding component in the form '`<profileName>.<stereotypeName>`'. The length of `stereotypes` must be the same as `compNames`.

Data Types: `string`

Output Arguments

components — Created components

array of components

Array of created components. This array is the same size as `compNames` and `stereotypes`.

See Also

`addPort` | `connect`

Topics

"Components"

Introduced in R2019a

addElement

Add a signal interface element

Syntax

```
element = addElement(interface,name)
element = addElement(interface,name,Name,Value)
```

Description

`element = addElement(interface,name)` adds an element to a signal interface with default properties.

`element = addElement(interface,name,Name,Value)` sets the properties of the element as specified in `Name,Value`.

Examples

Add an Interface and an Element

Add an interface `newinterface` to the interface dictionary of the model and add an element with type `double` to it.

```
interface = addInterface(archModel.InterfaceDictionary,'newsignal');
element = addElement(interface,'newelement','Type','double')
```

Input Arguments

interface — new interface object

signal interface

This is the interface that the new element is to be added.

Data Types: `systemcomposer.interface.SignalInterface`

name — Name of the new element

string

The new element name must be a valid variable name.

Data Types: char

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, . . . , NameN, ValueN`.

Example: `'Type', 'double'`

Type — Type of element

valid data type string

Data type of the element. Must be a valid data type.

Data Types: char

Dimensions — Dimensions of element

positive integer array

Each element is the size of the element in the corresponding direction. A scalar integer indicates a scalar or vector element, a row vector with two integers indicates a matrix element.

Data Types: char

Complexity — Complexity of element

real | complex

This describes whether the element is purely real, or if an imaginary part is allowed.

Data Types: string

Output Arguments

element — new interface element object

signal element

See Also

`getElement` | `getInterfaces` | `linkDictionary` |
`systemcomposer.createDictionary` | `unlinkDictionary`

Topics

“Define Interfaces”

Introduced in R2019a

addPort

Add ports to architecture

Syntax

```
ports = addPort(architecture, portNames, portTypes)
ports = addPort(architecture, portNames, portTypes, stereotypes)
```

Description

`ports = addPort(architecture, portNames, portTypes)` adds a set of ports with specified names.

`ports = addPort(architecture, portNames, portTypes, stereotypes)` also applies stereotypes.

Examples

Add Ports to Architecture

Create model, get root architecture, add component, and add ports.

```
model = systemcomposer.createModel('archModel');
rootArch = get(model, 'Architecture');
newcomponent = addComponent(rootArch, 'NewComponent');
newport = addPort(newcomponent.Architecture, 'NewCompPort', 'in');
```

Input Arguments

architecture — Component architecture

Architecture

`addPort` adds ports to the architecture of a component. Use `<component>.Architecture` to access the architecture of a component.

Data Types: `systemcomposer.arch.Architecture`

portNames — Names of ports

cell array of strings

Port names must be unique within each component. If necessary, System Composer appends a number to the port name to ensure uniqueness. The size of `portNames`, `portTypes`, and `stereotypes` must be the same.

Data Types: `string`

portTypes — Port directions

cell array of strings

Port directions are given in a cell array. Each element is either `'in'` or `'out'`.

Data Types: `string`

stereotypes — Stereotypes to apply to the components

Array of stereotypes

Each stereotype in the array must either be a mixin stereotype or a port stereotype. The size of `portNames`, `portTypes`, and `stereotypes` must be the same.

Data Types: `systemcomposer.profile.Stereotype`

Output Arguments

ports — Created ports

Array of ports

See Also

`addComponent` | `connect` | `destroy` | `systemcomposer.arch.BasePort`

Topics

“Ports”

Introduced in R2019a

addInterface

Create a named interface in an interface dictionary

Syntax

```
interface = addInterface(dictionary,name)
interface = addInterface(dictionary,name,busObject)
```

Description

`interface = addInterface(dictionary,name)` creates a named interface in the interface dictionary.

`interface = addInterface(dictionary,name,busObject)` constructs an interface that mirrors an existing Simulink® bus object.

Examples

Add an Interface

Add an interface `newinterface` to the interface dictionary of the model.

```
addInterface(archModel.InterfaceDictionary,'newinterface')
```

Input Arguments

dictionary — Data dictionary attached to the architecture model

System Composer dictionary

`dictionary` can be the default data dictionary that defines local interfaces or an external data dictionary that carries interface definitions. If the model links to multiple data dictionaries, then `dictionary` must be the one that carries interface definitions.

Data Types: `systemcomposer.interface.Dictionary`

name — **Name of the new interface**

`string`

The name of the new interface must be a valid variable name.

Data Types: `char`

busObject — **Simulink bus object that the new interface mirrors**

`Simulink bus`

Use this argument when the interface is already defined in a Simulink Bus object.

Data Types: `simulink bus`

Output Arguments

interface — **new interface object**

`signal interface`

Interface object with properties `Dictionary`, `Name`, and `Elements`.

See Also

`addElement` | `getInterface` | `getInterfaces` | `linkDictionary` | `systemcomposer.createDictionary`

Topics

“Define Interfaces”

Introduced in R2019a

addProperty

Add a property to a stereotype

Syntax

```
property = addProperty(stereotype, name, Name, Value)
```

Description

`property = addProperty(stereotype, name, Name, Value)` adds a new property with the specified `Name`, `Value` attributes.

Examples

Add a Property

Add a component stereotype and add a `VoltageRating` property with value 5.

```
stype = addStereotype(profile, 'electricalComponent', 'AppliesTo', 'Component')  
property = addProperty(stype, 'VoltageRating', 'DefaultValue', '5');
```

Input Arguments

stereotype — Stereotype to which the property is added

stereotype

name — Name of the property

string

Name of the property must be unique within the stereotype.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: `'Datatype', 'double'`

Datatype — Property data type

valid data type string

Data Types: char

Dimensions — Dimensions of property

positive integer array

Data Types: char

Min — Minimum value

numeric value

Data Types: double

Max — Maximum value

numeric value

Data Types: double

Units — Property units

string

Data Types: char

DefaultValue — Default value

numeric value

Data Types: double

Output Arguments

property — Created property

property

See Also

getProperty | setProperty

Topics

“Define Profiles and Stereotypes”

“Set Tags and Properties for Analysis”

Introduced in R2019a

addStereotype

Add a stereotype to the profile

Syntax

```
stereotype = addStereotype(profile, stereotypeName)  
stereotype = addStereotype(profile, stereotypeName, Name, Value)
```

Description

`stereotype = addStereotype(profile, stereotypeName)` adds a new stereotype with the specified name.

`stereotype = addStereotype(profile, stereotypeName, Name, Value)` specifies the properties of the stereotype.

Examples

Add a Component Stereotype

Add a component stereotype to the profile.

```
addStereotype(profile, 'electricalComponent', 'AppliesTo', 'Component')
```

Input Arguments

profile — Profile object

profile

The profile that contains the new stereotype.

Data Types: `systemcomposer.profile.Profile`

stereotypeName — Name of new stereotype

string

The name of the stereotype must be unique within the profile.

Data Types: char

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: 'AppliesTo', 'Component'

Name, Value — Stereotype properties and values

positive integer array

See `systemcomposer.profile.Stereotype` for stereotype properties and values.

Output Arguments

stereotype — Created stereotype

stereotype

See Also

`applyStereotype` | `removeStereotype`

Topics

“Define Profiles and Stereotypes”

Introduced in R2019a

applyProfile

Apply profile to a model

Syntax

```
applyProfile(modelObject,profileFile)
```

Description

`applyProfile(modelObject,profileFile)` applies the profile to a model and makes all of the constituent stereotypes available.

Input Arguments

modelObject — Architecture model object

architecture model

Data Types: `systemcomposer.arch.Model`

profileFile — Profile file

string

Data Types: `string`

See Also

`createProfile` | `removeProfile`

Topics

“Define Profiles and Stereotypes”

Introduced in R2019a

applyStereotype

Apply a stereotype to a model element

Syntax

```
applyStereotype(element, stereotype)
```

Description

`applyStereotype(element, stereotype)` applies a stereotype to a model element.

Input Arguments

element — Architecture model element

architecture component | architecture port | architecture connector

The stereotype is applied to this component, port, or connector.

Data Types: `systemcomposer.arch.Element`

stereotype — Reference stereotype

architecture stereotype

The qualified stereotype name in the form `<profile>.<stereotype>`. The profile must already be applied to the model.

Data Types: `char`

See Also

`applyStereotypeToAllComponents` | `applyStereotypeToAllConnections` | `applyStereotypeToAllPorts` | `removeStereotype`

Topics

“Use Stereotypes and Profiles”

Introduced in R2019a

applyStereotypeToAllComponents

Apply stereotype to all components in the specified architecture

Syntax

```
= applyStereotypeToAllComponents(architecture, stereotype)
= applyStereotypeToAllComponents(architecture, stereotype, true)
```

Description

= `applyStereotypeToAllComponents(architecture, stereotype)` applies the stereotype to all components within architecture.

= `applyStereotypeToAllComponents(architecture, stereotype, true)` applies the stereotype to all components within architecture and to their child components.

Input Arguments

architecture — Architecture model element

`architecture`

Parent architecture layer for all components to attach the stereotype.

Data Types: `systemcomposer.arch.Architecture`

stereotype — Stereotype to apply

`string`

Qualified name for the stereotype in the form '`profileName.stereotypeName`' The stereotype must be applicable to components.

Data Types: `string`

See Also

[applyStereotypeToAllConnections](#) | [applyStereotypeToAllPorts](#) | [removeStereotype](#)

Topics

"Use Stereotypes and Profiles"

Introduced in R2019a

applyStereotypeToAllConnections

Apply stereotype to all components in the specified architecture

Syntax

```
= applyStereotypeToAllConnections(architecture, stereotype)
= applyStereotypeToAllConnections(architecture, stereotype, true)
```

Description

= `applyStereotypeToAllConnections(architecture, stereotype)` applies the stereotype to all connections within architecture.

= `applyStereotypeToAllConnections(architecture, stereotype, true)` applies the stereotype to all connections within architecture and to connections in its child architectures, recursively.

Input Arguments

architecture — Architecture model element

architecture

Parent architecture layer for all connections to attach the stereotype.

Data Types: `systemcomposer.arch.Architecture`

stereotype — Stereotype to apply

string

Qualified name for the stereotype in the form '`profileName.stereotypeName`' The stereotype must be applicable to connections.

Data Types: `string`

See Also

[applyStereotype](#) | [applyStereotypeToAllComponents](#) | [applyStereotypeToAllPorts](#) | [removeStereotype](#)

Topics

“Use Stereotypes and Profiles”

Introduced in R2019a

applyStereotypeToAllPorts

Apply stereotype to all ports in the specified architecture

Syntax

```
= applyStereotypeToAllPorts(architecture, stereotype)
= applyStereotypeToAllPorts(architecture, stereotype, true)
```

Description

= `applyStereotypeToAllPorts(architecture, stereotype)` applies the stereotype to all ports within architecture.

= `applyStereotypeToAllPorts(architecture, stereotype, true)` applies the stereotype to all components within architecture and to their child components.

Input Arguments

architecture — Architecture model element

`architecture`

Parent architecture layer for all ports to attach the stereotype.

Data Types: `systemcomposer.arch.Architecture`

stereotype — Stereotype to apply

`string`

Qualified name for the stereotype in the form '`profileName.stereotypeName`'. The stereotype must be applicable to ports.

Data Types: `string`

See Also

[applyStereotype](#) | [applyStereotypeToAllComponents](#) | [applyStereotypeToAllConnections](#) | [removeStereotype](#)

Topics

“Use Stereotypes and Profiles”

Introduced in R2019a

open

Open architecture model

Syntax

```
open(modelObject)
```

Description

`open(modelObject)` opens the specified model in the editor if it is not already open.

Examples

Create and Open a Model

```
model = systemcomposer.createModel('archModel');  
open(model)
```

Input Arguments

modelObject — Architecture model object

architecture model

Data Types: `systemcomposer.arch.Model`

See Also

`createModel`

Topics

“Create an Architecture Model”

Introduced in R2019a

connect

Connect pairs of components

Syntax

```
connectors = connect(architecture,srcPorts,destPorts,stereotypes,  
rule)  
connectors = connect(srcComponent,destComponent,stereotypes,rule)
```

Description

`connectors = connect(architecture,srcPorts,destPorts,stereotypes,rule)` connects pairs of ports in the architecture. `connectors = connect(srcComponent,destComponent,stereotypes,rule)`

Examples

Connect Components

Create model, get root architecture, add ports, and connect ports.

```
arch = systemcomposer.createModel('archModel');  
rootArch = get(model,'Architecture');  
newcomponents = addComponent(rootArch,names);  
names = {'Component1','Component2'}  
output1 = addPort(newcomponents(1).Architecture,'','OutputPort');  
input1 = addPort(newcomponents(2).Architecture,'InputPort','');  
connect(rootArch,output1, input1);
```

Input Arguments

architecture — Architecture model element

Architecture

Data Types: `systemcomposer.arch.Architecture`

srcPorts — Array of source ports

array of ports

`srcPorts` must be the same length as `destPorts` and must consist of all output ports.

Data Types: `systemcomposer.arch.Port`

destPorts — Array of destination ports

array of ports

`destPorts` must be the same length as `srcPorts` and must consist of all source ports.

Data Types: `systemcomposer.arch.Port`

srcComponent — Source component

architecture component

Data Types: `systemcomposer.arch.Component`

destComponent — Destination component

architecture component

Data Types: `systemcomposer.arch.Component`

stereotypes — Stereotypes to apply to the connections

Array of stereotypes

Data Types: `systemcomposer.profile.Stereotype`

rule — Rule to match ports for connection

'name' | 'number'

Data Types: `systemcomposer.arch.Component`

Output Arguments

connectors — Created connections

Array of connections

See Also

addPort

Topics

“Create an Architecture Model”

Introduced in R2019a

systemcomposer.createDictionary

Create data dictionary

Syntax

```
dict_id = systemcomposer.createDictionary(dictionaryName)
```

Description

`dict_id = systemcomposer.createDictionary(dictionaryName)` creates a new Simulink data dictionary to hold interfaces and return a handle.

Input Arguments

dictionaryName — Name of new data dictionary

string

The name must include the `.sldd` extension

Example: `'new_dictionary.sldd'`

Data Types: char

Output Arguments

dictionary_id — Handle to the dictionary

dictionary object

Examples

```
dict_id = systemcomposer.createDictionary('new_dictionary.sldd')
```

See Also

`addInterface` | `linkDictionary` | `save` | `unlinkDictionary`

Topics

“Save and Link Interfaces”

Introduced in R2019a

createModel

Create a System Composer model

Syntax

```
model = systemcomposer.createModel(modelName)
```

Description

`model = systemcomposer.createModel(modelName)` creates a model with name `modelName` and returns its handle. Specify the optional `openFlag` as `TRUE` to open the model after creation.

Input Arguments

modelName — Name of new model

string

Model name must be a valid variable name.

Example: 'newModel'

Data Types: char | string

Output Arguments

model — Model handle

Architecture model object

Examples

```
m = systemcomposer.createModel('new_arch')
```

```
m =
```

```
Model with properties:
```

```
    Name: 'new_arch'  
    Architecture: [1x1 systemcomposer.arch.Architecture]  
    SimulinkHandle: 8.0001  
    Profiles: [1x1 systemcomposer.profile.Profile]
```

See Also

[loadModel](#) | [open](#)

Topics

“Compose Architecture Visually”

Introduced in R2019a

createProfile

Create profile

Syntax

```
profile = systemcomposer.createProfile(profileName,dirPath)
```

Description

`profile = systemcomposer.createProfile(profileName,dirPath)` creates a new profile object of type `systemcomposer.profile.Profile` to setup a set of stereotypes. The optional `dirPath` argument specifies a directory in which the profile is to be created.

Input Arguments

profileName — Name of new profile

string

Example: 'new_profile'

Data Types: char | string

Complex Number Support: No

Output Arguments

profile — Profile handle

profile object

Examples

```
systemcomposer.createProfile('new_profile')  
profile = systemcomposer.createProfile('new_profile')
```

See Also

`applyProfile` | `removeProfile` | `systemcomposer.loadProfile`

Topics

“Create a Profile and Add Stereotypes”

Introduced in R2019a

createSimulinkBehavior

Create a Simulink model and link component to it

Syntax

```
createSimulinkBehavior(component, modelName)
```

Description

`createSimulinkBehavior(component, modelName)` creates a new Simulink model with the same interface as the component and links the component to the new model. This method works only if the component has no children.

Examples

Create a Simulink Model and Link

Create a Simulink behavior model for the component `robotcomp` in `Robot.slx` and link the component to the model.

```
createSimulinkBehavior(robotcomp, 'Robot');
```

Input Arguments

component — Architecture component

architecture component

The component must have no children.

Data Types: `systemcomposer.arch.Component`

modelName — Model name

string

Name of the Simulink model created by this function.

Data Types: char

See Also

`linkToModel`

Topics

“Implement Components in Simulink”

Introduced in R2019a

deleteInstance

Delete an architecture instance

Syntax

```
deleteInstance(architectureInstance)
```

Description

deleteInstance(architectureInstance) deletes an existing instance.

Input Arguments

architectureInstance – The architecture instance

architecture instance

The architecture instance to be deleted.

Data Types: `systemcomposer.analysis.ArchitectureInstance`

See Also

instantiate

Topics

“Write Analysis Function”

Introduced in R2019a

destroy

Remove and destroy a model element

Syntax

```
destroy(element)
```

Description

`destroy(element)` removes and destroys the model element.

Examples

Destroy a Component

Create a component and then remove it from the model.

```
newcomponent = addComponent(rootArch, 'NewComponent');  
destroy(newcomponent)
```

Input Arguments

element — Architecture model element

architecture element | interface element | signal element | property

Data Types: `systemcomposer.arch.Element` |
`systemcomposer.interface.SignalInterface` |
`systemcomposer.interface.SignalElement` |
`systemcomposer.profile.Property`

See Also

`removeElement` | `removeProfile` | `removeProperty`

Introduced in R2019a

getActiveChoice

Get the active choice on the variant component

Syntax

```
choice = getActiveChoice(variantComponent)
```

Description

`choice = getActiveChoice(variantComponent)` finds which choice is active for the variant component.

Input Arguments

variantComponent — Architecture component
component

The architecture where the variant choices are selected.

Data Types: `systemcomposer.arch.Component`

Output Arguments

choice — Handle of chosen variant
component

Handle to the chosen variant.

Data Types: `systemcomposer.arch.Component`

See Also

`addChoice` | `getChoices` | `setActiveChoice`

Topics

“Create Variants”

Introduced in R2019a

getChoices

Get available choices in the variant component

Syntax

```
compList = getChoices(variantComponent)
```

Description

`compList = getChoices(variantComponent)` returns the list of choices available for a variant component.

Input Arguments

variantComponent — Architecture component
component

Variant component with multiple choices.

Data Types: `systemcomposer.arch.Component`

Output Arguments

compList — Choices available for the variant component
array of components

List of possible choices for the variant component.

See Also

`addChoice` | `getActiveChoice` | `setActiveChoice`

Topics

“Create Variants”

Introduced in R2019a

getCondition

Return the variant control on the choice within the variant component

Syntax

```
expression = getCondition(variantComponent,choice)
```

Description

`expression = getCondition(variantComponent,choice)` returns the variant control on the choice within the variant component.

Input Arguments

variantComponent — Architecture component
component

Variant component with multiple choices.

Data Types: `systemcomposer.arch.Component`

choice — Choice in a variant component
component

The choice whose control string is returned by this function.

Data Types: `systemcomposer.arch.Component`

Output Arguments

expression — The control string
string

The control string that controls the selection of the particular choice.

See Also

makeVariant | setActiveChoice | setCondition

Topics

“Create Variants”

Introduced in R2019a

getElement

Get the object a signal interface element

Syntax

```
element = getElement(interface,elementName)
```

Description

`element = getElement(interface,elementName)` gets the object for an element in a signal interface.

Examples

Get the Object for a Named Element

Add an interface `newinterface` to the interface dictionary of the model and add an element with type `double` to it. Then get the object for the element.

```
interface = addInterface(arch.InterfaceDictionary, 'newsignal');
addElement(interface, 'newelement', 'Type', 'double')
element = getElement(interface, 'newsignal')
element =
    SignalElement with properties:
```

```
    Interface: [1x1 systemcomposer.interface.SignalInterface]
        Name: 'newelement2'
        Type: 'double'
    Dimensions: '1'
        Units: ''
    Complexity: 'real'
        Minimum: '[]'
        Maximum: '[]'
    Description: ''
```

```
        UUID: 'f42c8166-e4ad-4488-926a-293050016e1a'  
ExternalUID: ''
```

Input Arguments

interface — interface object

signal interface

The object handle to the element to be identified.

Data Types: `systemcomposer.interface.SignalInterface`

elementName — Name of the element to be identified

string

Data Types: `char`

Output Arguments

element — new interface element object

signal element

See Also

`addElement` | `getInterface` | `removeElement`

Topics

“Define Interfaces”

Introduced in R2019a

getInterface

Get the object for a named interface in an interface dictionary

Syntax

```
interface = getInterface(dictionary,name)
```

Description

`interface = getInterface(dictionary,name)` gets the object for a named interface in the interface dictionary.

Examples

Add an Interface

Add an interface `newinterface` to the interface dictionary of the model. Obtain the interface object

```
addInterface(arch.InterfaceDictionary,'newsignal')
iface = getInterface(arch.InterfaceDictionary,'newsignal')
iface =
  SignalInterface with properties:
    Dictionary: [1x1 systemcomposer.interface.Dictionary]
      Name: 'newsignal'
    Elements: [0x0 systemcomposer.interface.SignalElement]
      UUID: '438b5004-6cab-40eb-955b-37e0df5a914f'
    ExternalUID: ''
```

Input Arguments

dictionary — Data dictionary

System Composer dictionary

This is the data dictionary attached to the model. It could be the local dictionary of the model or an external data dictionary.

Data Types: `systemcomposer.interface.Dictionary`

name — Name of the interface

string

Data Types: char

Output Arguments

interface — object for the interface

signal interface

See Also

`addElement` | `addInterface` | `removeElement`

Topics

“Define Interfaces”

Introduced in R2019a

getInterfaces

Get the object for a named interface in an interface dictionary

Syntax

```
interfaceList = getInterfaces(dictionary)
```

Description

`interfaceList = getInterfaces(dictionary)` gets the list of objects in the interface dictionary.

Examples

Get Interface List

```
ifaceList = getInterfaces(arch.InterfaceDictionary)
```

Input Arguments

dictionary — Data dictionary

System Composer dictionary

This is the data dictionary attached to the model. It could be the local dictionary of the model or an external data dictionary.

Data Types: `systemcomposer.interface.Dictionary`

Output Arguments

interfaceList — interface object list

array of signal interfaces

See Also

`addInterface` | `getInterface`

Topics

“Define Interfaces”

Introduced in R2019a

getProperty

Get the property value corresponding to a stereotype applied to the element

Syntax

```
[propertyValue,propertyUnits] = getProperty(element,propertyName)
```

Description

```
[propertyValue,propertyUnits] = getProperty(element,propertyName)
```

obtains the value and units of the property specified in the `propertyName` argument.

Examples

Get a Property from a Component

Get the weight property from a component with `sysComponent` stereotype applied.

```
>> [val, units] = getProperty(element, 'sysComponent.weight')
val =
    '0'
units =
    'kg'
```

Input Arguments

element — Architecture model element

architecture component | architecture port | architecture connector

This function gets the specified property of this element. A stereotype with the property must be applied to the element.

Data Types: `systemcomposer.arch.Element` |
`systemcomposer.arch.Architecture` | `systemcomposer.arch.Component` |
`systemcomposer.arch.Port`

propertyName — Name of the property

string

The property name must be qualified with the stereotype name, in the form '`<stereotype>.<property>`'.

Data Types: char

Output Arguments

propertyValue — Value of the property

string | number | enumeration

Data Types: char

propertyUnits — Unit of the property

string

Data Types: char

See Also

setProperty

Topics

“Set Tags and Properties for Analysis”

Introduced in R2019a

getValue

Get value of a property from an element instance

Syntax

```
[value,unit] = getValue(instance,property)
```

Description

`[value,unit] = getValue(instance,property)` obtains the property of the instance and assigns it to `value`. This function is part of the instance API that you can use to analyze the model iteratively, element by element. `instance` refers to the element instance on which the iteration is being performed.

Examples

Get the Weight Property

Assume that a `MechComponent` stereotype is attached to the specification of the instance.

```
weightValue = getValue(instance, 'MechComponent.weight');
```

Input Arguments

instance — The element instance

architecture instance | component instance | port instance | connector instance

This function is part of the instance API that you can use to analyze the model iteratively, element by element. `instance` refers to the element instance on which the iteration is being performed.

Data Types: `systemcomposer.analysis.ArchitectureInstance` |
`systemcomposer.analysis.ComponentInstance` |
`systemcomposer.analysis.PortInstance` |
`systemcomposer.analysis.ConnectorInstance`

property — The property field

`stereotype.property`

String in the form `<stereotype>.<property>`.

Data Types: `string`

Output Arguments

value — Property value

any variable type

Value of the property. The data type depends on how the property is defined in the profile.

unit — Property unit

`string`

String that describe the unit of the property as defined in the profile.

See Also

`setValue`

Topics

“Write Analysis Function”

Introduced in R2019a

inlineComponent

Inline reference architecture into model

Syntax

```
componentHandle = inlineComponent(component,inlineFlag)
```

Description

`componentHandle = inlineComponent(component,inlineFlag)` inlines the contents of the architecture model referenced by the specified `component` and breaks the link to the reference model. If `inlineFlag` is `false`, then the contents are removed and only interfaces remain.

Examples

Reuse a Component

Save the component `robotcomp` in the architecture model `Robot.slx` and reference it from another component, `robotArm` so that `robotArm` uses the architecture of `robotcomp`. Inline `robotcomp` so that its architecture can be edited independently.

```
saveAsModel(robotcomp, 'Robot');  
linkToModel(robotArm, 'Robot');  
inlineComponent(robotArm,true);
```

Input Arguments

component — Architecture component

architecture component

The component must be linked to an architecture model.

Data Types: `systemcomposer.arch.Component`

inlineFlag — control the contents of the inlined component

true | false

If `true`, contents of the referenced architecture model are copied to the component architecture. If `false`, the contents are not copied, only ports and interfaces are inlined.

Data Types: `char`

Output Arguments

componentHandle — Component object

architecture component

See Also

`saveAsModel`

Topics

“Decompose and Reuse Components”

Introduced in R2019a

instantiate

Create an analysis instance from a specification

Syntax

```
instance = instantiate(model,properties,name)
```

Description

`instance = instantiate(model,properties,name)` creates an instance of a model for analysis.

Input Arguments

model — Handle to the model

`model handle`

The instance is generated from the model specified in this argument.

properties — Stereotype properties which require values in the instance model

`instance properties object`

Each value for an instance in an instance model can be drawn from any stereotype in any profile on the path. The structure of the property definition parameter accommodates this approach. The definition is a structure with a field for each profile of interest. The name of the field is the name of the profile. Each profile field is itself a structure, which has a field per stereotype whose name is the name of the stereotype. Each stereotype in turn is another structure that contains two fields, one called `properties`, which specifies properties of interest and another called `elementKinds` which indicates the kinds of instance to which the values corresponding to the properties are added. The `properties` field is a structure that lists the required properties as Boolean fields; the name of the field is the name of the property and the value indicates whether the field can be set via the API. The `elementKinds` field is a list of strings whose value must be one of: 'Component', 'Port' or 'Connector' to indicate the applicable elements.

Data Types: `systemcomposer.analysis.InstanceProperties`

name — Name of the instance

string

This is the name given to the instance generated from the model.

Output Arguments

instance — The element instance

architecture instance | component instance | port instance | connector instance

This function is part of the instance API that you can use to analyze the model iteratively, element by element. `instance` refers to the element instance on which the iteration is being performed.

Data Types: `systemcomposer.analysis.ArchitectureInstance`

See Also

`deleteInstance` | `loadInstance` | `saveInstance`

Topics

“Write Analysis Function”

Introduced in R2019a

isArchitecture

Find if an instance is a architecture instance

Syntax

```
flag = isComponent(instance)
```

Description

`flag = isComponent(instance)` finds whether the instance is a architecture instance.

Input Arguments

instance — The element instance

architecture instance | component instance | port instance | connector instance

This function is part of the instance API that you can use to analyze the model iteratively, element by element. `instance` refers to the element instance on which the iteration is being performed.

Data Types: `systemcomposer.analysis.ArchitectureInstance` |
`systemcomposer.analysis.ComponentInstance` |
`systemcomposer.analysis.PortInstance` |
`systemcomposer.analysis.ConnectorInstance`

Output Arguments

flag — Indicate if the instance is a architecture

boolean

This argument is `true` if the instance is a architecture.

See Also

isComponent | isConnector | isPort

Topics

“Write Analysis Function”

Introduced in R2019a

isComponent

Find if an instance is a component instance

Syntax

```
flag = isComponent(instance)
```

Description

`flag = isComponent(instance)` finds whether the instance is a component instance.

Input Arguments

instance — The element instance

architecture instance | component instance | port instance | connector instance

This function is part of the instance API that you can use to analyze the model iteratively, element by element. `instance` refers to the element instance on which the iteration is being performed.

Data Types: `systemcomposer.analysis.ArchitectureInstance` |
`systemcomposer.analysis.ComponentInstance` |
`systemcomposer.analysis.PortInstance` |
`systemcomposer.analysis.ConnectorInstance`

Output Arguments

flag — Indicate if the instance is a component

boolean

This argument is `true` if the instance is a component.

See Also

isArchitecture | isConnector | isPort

Topics

“Write Analysis Function”

Introduced in R2019a

isConnector

Find if an instance is a connector instance

Syntax

```
flag = isConnector(instance)
```

Description

`flag = isConnector(instance)` finds whether the instance is a connector instance.

Input Arguments

instance — The element instance

architecture instance | component instance | port instance | connector instance

This function is part of the instance API that you can use to analyze the model iteratively, element by element. `instance` refers to the element instance on which the iteration is being performed.

Data Types: `systemcomposer.analysis.ArchitectureInstance` |
`systemcomposer.analysis.ComponentInstance` |
`systemcomposer.analysis.PortInstance` |
`systemcomposer.analysis.ConnectorInstance`

Output Arguments

flag — Indicate if the instance is a connector

boolean

This argument is `true` if the instance is a connector.

See Also

isArchitecture | isComponent | isPort

Topics

“Write Analysis Function”

Introduced in R2019a

isPort

Find if an instance is a port instance

Syntax

```
flag = isPort(instance)
```

Description

`flag = isPort(instance)` finds whether the instance is a port instance.

Input Arguments

instance — The element instance

architecture instance | component instance | port instance | connector instance

This function is part of the instance API that you can use to analyze the model iteratively, element by element. `instance` refers to the element instance on which the iteration is being performed.

Data Types: `systemcomposer.analysis.ArchitectureInstance` |
`systemcomposer.analysis.ComponentInstance` |
`systemcomposer.analysis.PortInstance` |
`systemcomposer.analysis.ConnectorInstance`

Output Arguments

flag — Indicate if the instance is a port

boolean

This argument is `true` if the instance is a port.

See Also

isArchitecture | isConnector | isConnector

Topics

“Write Analysis Function”

Introduced in R2019a

linkDictionary

Link data dictionary to an architecture model

Syntax

```
linkDictionary(modelObject,dictionaryFile)
```

Description

`linkDictionary(modelObject,dictionaryFile)` associates the specified Simulink Data Dictionary with the model.

Input Arguments

modelObject — Architecture model object

Data Types: `systemcomposer.arch.Model`

dictionaryFile — Dictionary file name with the `.sldd` extension
string

Data Types: `string`

See Also

`getInterfaces` | `systemcomposer.createDictionary`

Topics

“Save and Link Interfaces”

Introduced in R2019a

linkToModel

Link component to a model

Syntax

```
modelHandle = linktoModel(component, modelName)
```

Description

`modelHandle = linktoModel(component, modelName)` links from the component to a model.

Examples

Reuse a Component

Save the component `robotcomp` in the architecture model `Robot.slx` and reference it from another component, `robotArm` so that `robotArm` uses the architecture of `robotcomp`.

```
saveAsModel(robotcomp, 'Robot');  
linkToModel(robotArm, 'Robot');
```

Input Arguments

component — Architecture component

architecture component

The component must have no children.

Data Types: `systemcomposer.arch.Component`

modelName — Model name

string

An existing model that define the architecture or behavior of the component.

Data Types: char

Output Arguments

modelHandle — Handle to the linked model

numeric handle

See Also

inlineComponent

Topics

“Decompose and Reuse Components”

Introduced in R2019a

loadInstance

Load an architecture instance

Syntax

```
loadInstance(fileName,overwrite)
```

Description

loadInstance(fileName,overwrite) loads an architecture instance from a MAT-file.

Input Arguments

fileName — File that contains an architecture instance

string

This is a MAT-file that was previously saved with an architecture instance.

overwrite — Whether to overwrite an instance if it already exists in the workspace

1 | 0

If true, the load operation overwrites duplicate instances in the workspace.

See Also

deleteInstance | saveInstance | updateInstance

Topics

“Write Analysis Function”

Introduced in R2019a

loadModel

Load architecture model

Syntax

```
model = systemcomposer.loadModel(modelName)
```

Description

`model = systemcomposer.loadModel(modelName)` loads the model with name `modelName` and returns its handle. The loaded model is not displayed.

Input Arguments

modelName — Name of model

string

Model must exist on the MATLAB® path.

Example: 'new_arch'

Data Types: char | string

Output Arguments

model — Model handle

Model object

Examples

```
systemcomposer.loadModel('new_arch')  
model = systemcomposer.loadModel('new_arch')
```


See Also

open | save

Topics

“Create an Architecture Model”

Introduced in R2019a

systemcomposer.loadProfile

Load profile

Syntax

```
profile = systemcomposer.loadProfile(profileName)
```

Description

`profile = systemcomposer.loadProfile(profileName)` loads a profile with the specified file name

Input Arguments

profileName — Name of new profile

string

Profile must be available on the MATLAB path.

Example: 'new_profile'

Data Types: char | string

Output Arguments

profile — Profile handle

Profile object

Examples

```
systemcomposer.loadProfile('new_profile')  
profile = systemcomposer.loadProfile('new_profile')
```

See Also

applyProfile

Topics

“Define Profiles and Stereotypes”

Introduced in R2019a

lookup

Lookup an architecture element

Syntax

```
lookup(modelObject,Name,Value)
```

Description

lookup(modelObject,Name,Value) finds an architecture element based in its UUID or full path.

Examples

Look up a Component by Path

```
>> lookup(arch, 'Path', 'RobotSystem/Sensors')
```

```
ans =
```

```
Component with properties:
```

```
    Name: 'Sensors'  
    Parent: [1x1 systemcomposer.arch.Architecture]  
    Ports: [1x2 systemcomposer.arch.ComponentPort]  
    OwnedPorts: []  
    Architecture: [1x1 systemcomposer.arch.Architecture]  
    OwnedArchitecture: []  
    Position: [275 75 391 161]  
    Model: [1x1 systemcomposer.arch.Model]  
    UUID: 'f43c9d51-9dc6-43fc-b3af-95d458b81248'  
    SimulinkHandle: 9.0002
```

```
SimulinkModelHandle: 2.0002  
ExternalUID: ''
```

Input Arguments

modelObject — Architecture model object

Data Types: `systemcomposer.arch.Model`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: `'Path', 'RobotSystem/Sensors'`

UUID — UUID of the element

character vector

Data Types: `char`

Path — Path to the element

character vector

Path to the model element, specified as a character vector.

Data Types: `char`

SimulinkHandle — Simulink handle of the element

double

Simulink handle of the element

Data Types: `double`

See Also

`instantiate`

Topics

“Analyze Architecture”

Introduced in R2019a

makeVariant

Convert component to a variant choice

Syntax

```
[variantComp,choices] = makeVariant(components)
```

Description

[variantComp,choices] = makeVariant(components) converts components to variant choices and returns the parent component and available choices.

Input Arguments

components — Architecture components

array of components

Architecture components to be converted to variants.

Data Types: `systemcomposer.arch.Component`

Output Arguments

variantComp — Component containing the variants

component

Component that contains the variants.

choices — Variant choice names

cell array of strings

Choices available in the new variant.

Data Types: `string`

See Also

addChoice | getChoices

Topics

“Create Variants”

Introduced in R2019a

systemcomposer.openModel

Open a System Composer architecture model

Syntax

```
model = systemcomposer.openModel(modelName)
```

Description

`model = systemcomposer.openModel(modelName)` opens the model with name `modelName` for editing and returns its handle.

Input Arguments

modelName — Name of new model

string

Model must exist on the MATLAB path.

Example: 'new_arch'

Data Types: char | string | Model

Output Arguments

model — Model handle

Model object

Examples

```
systemcomposer.openModel('new_arch')  
model = systemcomposer.openModel('new_arch')
```

See Also

`createModel` | `open`

Topics

“Create an Architecture Model”

Introduced in R2019a

removeElement

Remove a signal interface element

Syntax

```
removeElement(interface,elementName)
```

Description

`removeElement(interface,elementName)` removes an element from a signal interface.

Examples

Add an Interface and an Element

Add an interface `newinterface` to the interface dictionary of the model and add an element with type `double` to it, then remove the element.

```
interface = addInterface(arch.InterfaceDictionary, 'newsignal');  
element = addElement(interface, 'newelement', 'Type', 'double');  
removeElement(interface, 'newsignal')
```

Input Arguments

interface — interface object

signal interface

Data Types: `systemcomposer.interface.SignalInterface`

elementName — Name of the element to be removed

String

Data Types: `char`

See Also

addElement | getElement

Topics

“Define Interfaces”

Introduced in R2019a

removeInterface

Remove a named interface from an interface dictionary

Syntax

```
removeInterface(dictionary,name)
```

Description

`removeInterface(dictionary,name)` removes a named interface from the interface dictionary.

Examples

Remove an Interface

Add an interface `newinterface` to the interface dictionary of the model and then remove it.

```
addInterface(arch.InterfaceDictionary,'newsignal')  
removeInterface(arch.InterfaceDictionary,'newsignal')
```

Input Arguments

dictionary — Data dictionary attached to the architecture model

System Composer dictionary

Data Types: `systemcomposer.interface.Dictionary`

name — Name of the new interface

string

Data Types: `char`

See Also

`addInterface` | `getInterface` | `getInterfaces`

Topics

“Define Interfaces”

Introduced in R2019a

removeProfile

Remove profile from a model

Syntax

```
removeProfile(modelObject,profileFile)
```

Description

`removeProfile(modelObject,profileFile)` applies the profile to a model and makes all of the constituent stereotypes available.

Examples

Remove a Profile

```
removeProfile(arch,'SystemProfile')
```

Input Arguments

modelObject — Architecture model object

architecture model

Data Types: `systemcomposer.arch.Model`

profileFile — Profile file

string

Name of a profile attached to the model.

Data Types: `string`

See Also

`applyProfile` | `createProfile`

Topics

“Define Profiles and Stereotypes”

Introduced in R2019a

removeProperty

Remove a property from a stereotype

Syntax

```
removeProperty(stereotype,propertyName)
```

Description

`removeProperty(stereotype,propertyName)` removes a property from the stereotype.

Examples

Remove a Property

Add a component stereotype and add a `VoltageRating` property with value 5. Then remove the property.

```
styp = addStereotype(profile,'electricalComponent','AppliesTo','Component')
property = addProperty(styp,'VoltageRating','DefaultValue','5');
removeProperty(styp,'VoltageRating');
```

Input Arguments

stereotype — Stereotype to which the property is added

stereotype

propertyName — Property to be removed

string

See Also

addProperty

Topics

“Define Profiles and Stereotypes”

Introduced in R2019a

removeStereotype

Remove a stereotype from a model element

Syntax

```
removeStereotype(element, stereotype)
```

Description

`removeStereotype(element, stereotype)` removes a stereotype from the mode element.

Input Arguments

element — Architecture model element

architecture component | architecture port | architecture connector

The stereotype and all its properties are removed from this element.

Data Types: `systemcomposer.arch.Element`

stereotype — Reference stereotype

stereotype

The stereotype must be specified in the form `<profile>.<stereotype>`.

Data Types: `systemcomposer.internal.profile.Stereotype`

See Also

`applyStereotype`

Topics

“Remove a Stereotype”

Introduced in R2019a

reparent

Move stereotype

Syntax

```
reparent(stereotype, parentStereotype)
```

Description

`reparent(stereotype, parentStereotype)` reparents the stereotype to the specified stereotype.

Examples

Reparent a Property

Add an architecture stereotype and reparent it to a component.

```
styp = addStereotype(profile, 'electricalComponent', 'systemcomposer.Architecture', 'General')
reparent(styp, 'systemcomposer.Component')
```

Input Arguments

stereotype — Stereotype whose inheritance changes

stereotype

parentStereotype — the new stereotype to inherit from

stereotype

See Also

Introduced in R2019a

save

Save the architecture model or data dictionary

Syntax

```
save(architecture)
save(dictionary)
```

Description

`save(architecture)` saves the architecture model to the file specified in its `Name` property.

`save(dictionary)` saves the data dictionary.

Examples

Save Model and Data Dictionary

```
save(arch);
save(arch.InterfaceDictionary);
```

Input Arguments

architecture — The architecture model

System Composer architecture

Data Types: `systemcomposer.arch.Model`

dictionary — Data dictionary attached to the architecture model

System Composer dictionary

Data Types: `systemcomposer.interface.Dictionary`

See Also

loadModel

Topics

“Create an Architecture Model”

“Save and Link Interfaces”

Introduced in R2019a

saveAsModel

Save the Architecture to a separate model

Syntax

```
saveAsModel ( component , modelName )
```

Description

`saveAsModel (component , modelName)` saves the architecture of the component to a separate architecture model and references the model from this component.

Examples

Save a Component

Save the component `robotcomp` in `Robot.slx` and reference the model.

```
saveAsModel ( robotcomp , 'Robot' );
```

Input Arguments

component — Architecture component

architecture component

The component must have an architecture with definition type `composition`. For other definition types, this function gives an error.

Data Types: `systemcomposer.arch.Component`

modelName — Model name

string

Data Types: char

See Also

`inlineComponent` | `linkToModel`

Topics

“Decompose and Reuse Components”

Introduced in R2019a

saveInstance

Save an architecture instance

Syntax

```
saveInstance(architectureInstance, fileName)
```

Description

saveInstance(architectureInstance, fileName) saves an architecture instance to a MAT-file.

Input Arguments

architectureInstance — The architecture instance

architecture instance

The architecture instance to be saved.

Data Types: `systemcomposer.analysis.ArchitectureInstance`

fileName — File to save the instance

string

This is a MAT-file to save the architecture instance.

See Also

loadInstance

Topics

“Write Analysis Function”

Introduced in R2019a

setActiveChoice

Set the active choice in the variant component

Syntax

```
setActiveChoice(variantComponent, choice)
```

Description

`setActiveChoice(variantComponent, choice)` sets the active choice on the variant component.

Input Arguments

variantComponent — **Architecture component**
component

Variant component with multiple choices.

Data Types: `systemcomposer.arch.Component`

choice — **Choice in a variant component**
component | string

The choice whose control string is returned by this function. This can be a component object or label of the variant choice.

Data Types: `systemcomposer.arch.Component` | `string`

See Also

`addChoice` | `getActiveChoice` | `getChoices`

Topics

“Create Variants”

Introduced in R2019a

setCondition

Set the condition on the variant choice

Syntax

```
setCondition(variantComponent,choice, expression)
```

Description

`setCondition(variantComponent,choice, expression)` sets the variant control for a choice for the variant component.

Input Arguments

variantComponent — Architecture component
component

Variant component with multiple choices.

Data Types: `systemcomposer.arch.Component`

choice — Choice in a variant component
component | string

The choice whose control string is set by this function.

Data Types: `systemcomposer.arch.Component`

expression — The control string
string

The control string that controls the selection of the choice.

See Also

`getCondition` | `makeVariant` | `setActiveChoice`

Topics

“Create Variants”

Introduced in R2019a

setProperty

Set the property value corresponding to a stereotype applied to the element

Syntax

```
setProperty(element, propertyName, propertyValue, propertyUnits)
```

Description

`setProperty(element, propertyName, propertyValue, propertyUnits)` sets the value and units of the property specified in the `propertyName` argument.

Examples

Apply a Stereotype and Set Numeric Property Value

In this example, `weight` is a property of the stereotype `sysComponent`.

```
applyStereotype(element, 'sysProfile.sysComponent')
setProperty(element, 'sysComponent.weight', '5', 'g')
```

Apply a Stereotype and Set String Property Value

In this example, `description` is a property of the stereotype `sysComponent`.

```
expression = sprintf("%s", 'component description')
setProperty(element, 'sysComponent.description', expression)
```

Input Arguments

element — Architecture model element

architecture component | architecture port | architecture connector

Data Types: `systemcomposer.arch.Element`

propertyName — Name of the property

`stereotype.property`

Qualified name of the property in the form '`<stereotype>.<property>`'.

Data Types: `char`

propertyValue — Value of the property

`string`

Specify numeric values in single quotes. Specify string values in the `printf("%s", '<property value>')` form. See example on this page.

Data Types: `char`

propertyUnits — Units of the property

`string`

Specify the units to interpret property values.

Data Types: `char`

See Also

`getProperty`

Topics

“Set Tags and Properties for Analysis”

Introduced in R2019a

setValue

Set the value of a property for an element instance

Syntax

```
setValue(instance,property,value)
```

Description

`setValue(instance,property,value)` sets the property of the instance to value. This function is part of the instance API that you can use to analyze the model iteratively, element by element.`instance` refers to the element instance on which the iteration is being performed.

Examples

Set the Weight Property

Assume that a `MechComponent` stereotype is attached to the specification of the instance.

```
setValue(instance,'MechComponent.weight',10);
```

Input Arguments

instance — The element instance

architecture instance | component instance | port instance | connector instance

This function is part of the instance API that you can use to analyze the model iteratively, element by element.`instance` refers to the element instance on which the iteration is being performed.

Data Types: `systemcomposer.analysis.ArchitectureInstance` |
`systemcomposer.analysis.ComponentInstance` |
`systemcomposer.analysis.PortInstance` |
`systemcomposer.analysis.ConnectorInstance`

property — The property field

`stereotype.property`

String in the form `<stereotype>.<property>`.

Data Types: `string`

See Also

`getValue`

Topics

“Write Analysis Function”

Introduced in R2019a

unlinkDictionary

Unlink dictionary from a model

Syntax

```
unlinkDictionary(modelObject)
```

Description

`unlinkDictionary(modelObject)` removes the association of the model from its data dictionary.

Examples

Unlink the Data Dictionary

```
unlinkDictionary(arch);
```

Input Arguments

modelObject — Architecture model object
`architecture`

The model from which the dictionary link is to be removed.

Data Types: `systemcomposer.arch.Model`

See Also

`linkDictionary`

Topics

“Save and Link Interfaces”

Introduced in R2019a

updateInstance

Update an architecture instance

Syntax

```
updateInstance(architectureInstance,updateFlag)
```

Description

`updateInstance(architectureInstance,updateFlag)` updates an instance to mirror the changes in the specification model.

Input Arguments

architectureInstance — The architecture instance

`architecture instance`

The architecture instance to be updated.

Data Types: `systemcomposer.analysis.ArchitectureInstance`

updateFlag — whether to update values changed directly in the model

`1 | 0`

If true, the method reflects changes made directly in the specification model to the instance model.

See Also

`loadInstance` | `saveInstance`

Topics

“Write Analysis Function”

Introduced in R2019a

Classes — Alphabetical List

systemcomposer.analysis.Instance

Class that represents an architecture model element in an analysis instance

Description

The Instance class represents an instance of an architecture.

Creation

Create an instance of an architecture

```
instance = instantiate(modelHandle,architecture,properties,name)
```

Properties

Name — Name of the instance

string

This is the name of the instance.

Data Types: char

Specification — The specification that the instance is created from

architecture | component | port | connector

Every instance has a specification from which it took its form. The kind of the specification depends on the kind of the instance.

Data Types: systemcomposer.arch.Architecture |
systemcomposer.arch.Component | systemcomposer.arch.Port |
systemcomposer.arch.Connector

Architecture Instance Properties

Components — Child components of the instance

array of components

The components within the architecture.

Data Types: `systemcomposer.analysis.ComponentInstance`

Ports — Ports of the architecture instance

array of ports

These are the architecture ports that belong to the architecture instance.

Data Types: `systemcomposer.analysis.PortInstance`

Connectors — Connectors in the architecture instance

array of connectors

These are the connectors within the architecture, connecting child components.

Data Types: `systemcomposer.analysis.Connectors`

Component Instance Properties

Components — Child components of the instance

array of components

The components within the architecture.

Data Types: `systemcomposer.analysis.ComponentInstance`

Ports — Ports of the architecture instance

array of ports

These are the architecture ports that belong to the architecture instance.

Data Types: `systemcomposer.analysis.PortInstance`

Connectors — Connectors in the architecture instance

array of connectors

These are the connectors within the architecture, connecting child components.

Data Types: `systemcomposer.analysis.Connectors`

Parent — Parent of the component

component

The architecture that contains the component

Data Types: `systemcomposer.analysis.Architecture`

Port Instance Properties

Parent — Parent of the port

`component`

The component that contains the port

Data Types: `systemcomposer.analysis.Component`

Connector Instance Properties

Parent — Parent of the connector

`component`

The component that contains the connector

Data Types: `systemcomposer.analysis.Component`

SourcePort — Source port

`port`

The port from which the connector originates.

Data Types: `systemcomposer.analysis.Port`

DestinationPort — Destination port

`port`

The port from which the connector ends.

Data Types: `systemcomposer.analysis.Port`

Object Functions

| | |
|-----------------------------|--|
| <code>deleteInstance</code> | Delete an architecture instance |
| <code>getValue</code> | Get value of a property from an element instance |
| <code>instantiate</code> | Create an analysis instance from a specification |
| <code>isArchitecture</code> | Find if an instance is a architecture instance |
| <code>isComponent</code> | Find if an instance is a component instance |

| | |
|----------------|---|
| isConnector | Find if an instance is a connector instance |
| isPort | Find if an instance is a port instance |
| loadInstance | Load an architecture instance |
| saveInstance | Save an architecture instance |
| setValue | Set the value of a property for an element instance |
| updateInstance | Update an architecture instance |

See Also

Topics

“Write Analysis Function”

Introduced in R2019a

systemcomposer.arch.Architecture

Class that represents an Architecture in the model

Description

The Architecture class represents an architecture in the model

Creation

Create an model and get the root architecture:

```
model = systemcomposer.createModel('archModel');  
arch=get(model, 'Architecture')
```

Properties

Name — Name of the architecture

string

The architecture name is derived from the parent component or model name to which the architecture belongs.

Example: 'system_architecture'

Data Types: char

Definition — Definition type of the architecture

Composition | Behavior | View

The definition type can be a composition, a behavior model, or a view.

Example: Composition

Data Types: ArchitectureDefinition enum

Parent — Handle to the parent component that owns this Architecture

Architecture component object

Data Types: `systemcomposer.arch.Component`

Components — Array of handles to the set of child components of this architecture

array of component objects

Data Types: `systemcomposer.arch.Component`

Ports — Array of architecture ports of this architecture

array of ports

Data Types: `systemcomposer.arch.ArchitecturePort`

Connectors — Array of connectors that either interconnect child components or connect child components to architecture ports

array of connectors

Data Types: `systemcomposer.arch.Connector`

Object Functions

`addComponent` Add a component to the architecture

`addPort` Add ports to architecture

`connect` Connect pairs of components

See Also

`systemcomposer.arch.Component`

Topics

“Create an Architecture Model”

Introduced in R2019a

systemcomposer.arch.BasePort

Base class of both architecture and component ports

Description

The `BasePort` class is the base class for all ports, both architecture ports and component ports. This class is derived from `systemcomposer.arch.Element`

Creation

Create a port.

`addPort`

Properties

Name — Name of port

`string`

Direction — Port direction

`'Input' | 'Output'`

Interface — Interface attached to the port

`signal interface`

Data Types: `systemcomposer.interface.SignalInterface`

Object Functions

`connect` Connect pairs of components

See Also

`systemcomposer.arch.Element`

Topics

“Ports”

Introduced in R2019a

systemcomposer.arch.Component

Class that represents a component or view component

Description

The Component class represents a component in the architecture model

Creation

Create a component in an architecture model:

```
model = systemcomposer.createModel('archModel');  
arch=get(model, 'Architecture');  
component = addComponent(arch, 'NewComponent');
```

Properties

ParentArchitecture — Handle to the parent component that owns this component

Architecture object

Data Types: `systemcomposer.arch.Architecture`

Architecture — Architecture that defines the component structure

Architecture object

For a component that references a different architecture model, this returns a handle to the root architecture of that model. For variant components, the architecture is that of the active variant.

Data Types: `systemcomposer.arch.Architecture`

OwnedArchitecture — The architecture that this component directly owns architecture

For components that reference an architecture, this is be empty. For variant components , this is the architecture in which the individual variant components reside.

Data Types: `systemcomposer.arch.Architecture`

Ports — Array of component ports

array of ports

Data Types: `systemcomposer.arch.ComponentPort`

OwnedPorts — Array of component ports

array of ports

For all components except Variant View components, this will return the same value as Ports. For Variant View components, this returns the aggregate of all ports across all Views in which this component is present.

Data Types: `systemcomposer.arch.ComponentPort`

ReferenceName — If linked component, the name of the model that the component references

string

Data Types: `char`

Object Functions

| | |
|-------------------------------------|--|
| <code>saveAsModel</code> | Save the Architecture to a separate model |
| <code>createSimulinkBehavior</code> | Create a Simulink model and link component to it |
| <code>linkToModel</code> | Link component to a model |
| <code>inlineComponent</code> | Inline reference architecture into model |
| <code>connect</code> | Connect pairs of components |

See Also

`systemcomposer.arch.Architecture`

Topics

“Create an Architecture Model”

Introduced in R2019a

systemcomposer.arch.Connector

Class that represents a connector between ports

Description

The connector class represents a connectore between ports. This class is derived from `systemcomposer.arch.element`

Creation

Create a connector.

```
connector = connect(architecture, outports, inports)
```

Properties

ParentArchitecture — Handle to the parent component that owns this component

Architecture object

Data Types: `systemcomposer.arch.Architecture`

SourcePort — Source of the connection

architecture port | component port

The source port is an output port.

DestinationPort — Destination of the connection

architecture port | component port

The destination port is an input port.

Direction — Port direction

'Input' | 'Output'

Interface — Interface attached to the port

signal interface

Data Types: `systemcomposer.interface.SignalInterface`

Object Functions

See Also

`systemcomposer.arch.Element`

Topics

“Create an Architecture Model”

Introduced in R2019a

systemcomposer.arch.Element

Base class of all model elements

Description

The `Element` class is the base class for all model elements — Architecture, component, port, and connector.

Creation

Create an architecture, component, port, or connector:

```
addComponent  
addPort  
connect
```

Properties

UUID — Unique identifier of the model element

string

Example: '91d5de2c-b14c-4c76-a5d6-5dd0037c52df'

Data Types: char

ExternalUUID — External identifier

string

Set an external ID that is preserved over the lifespan of the element. The external ID is preserved through all operations that preserve the UUID.

Example: 'network_connector_01'

Data Types: char

Model — Handle to the parent System Composer model of the element

architecture model object

Data Types: `systemcomposer.arch.Model`

Object Functions

| | |
|-------------------------------|---|
| <code>applyStereotype</code> | Apply a stereotype to a model element |
| <code>removeStereotype</code> | Remove a stereotype from a model element |
| <code>setProperty</code> | Set the property value corresponding to a stereotype applied to the element |
| <code>getProperty</code> | Get the property value corresponding to a stereotype applied to the element |
| <code>destroy</code> | Remove and destroy a model element |

See Also

`systemcomposer.arch.BasePort` | `systemcomposer.arch.Component` | `systemcomposer.arch.Connector`

Topics

“Create an Architecture Model”

Introduced in R2019a

systemcomposer.arch.Model

Represent a System Composer model

Description

The Model class is used to create and manage objects in the model

Creation

```
model = systemcomposer.createModel(Name)
```

Properties

Name — Name of the model

string

Example: 'archModel'

Data Types: char

Architecture — Root architecture of a System Composer model

systemcomposer.arch.Architecture

Data Types: systemcomposer.arch.Architecture

SimulinkHandle — Handle to the Simulink representation of the System Composer model

double number

Data Types: double

Profiles — Array of handles to profiles attached to the model

profiles array

Data Types: systemcomposer.internal.profile.Profile

InterfaceDictionary — The dictionary object that holds interfaces. If the model is not linked to an external dictionary, this is a handle to the implicit dictionary

dictionary object

Data Types: `systemcomposer.interface.Dictionary`

Object Functions

| | |
|-------------------------------|--|
| <code>open</code> | Open architecture model |
| <code>save</code> | Save the architecture model or data dictionary |
| <code>applyProfile</code> | Apply profile to a model |
| <code>removeProfile</code> | Remove profile from a model |
| <code>linkDictionary</code> | Link data dictionary to an architecture model |
| <code>unlinkDictionary</code> | Unlink dictionary from a model |
| <code>lookup</code> | Lookup an architecture element |

See Also

Topics

“Create an Architecture Model”

Introduced in R2019a

systemcomposer.interface.Dictionary

Class that represents an element in the signal interface

Description

The `systemcomposer.interface.Dictionary` class represents the interface dictionary of an architecture model.

Creation

Create a signal element.

```
dictionary = <architecture>.InterfaceDictionary;
```

Properties

Interfaces — Interfaces defined in the dictionary

array of signal interfaces

Data Types: `systemcomposer.interface.Dictionary`

UUID — Unique identifier

string

Object Functions

| | |
|------------------------------|---|
| <code>addInterface</code> | Create a named interface in an interface dictionary |
| <code>removeInterface</code> | Remove a named interface from an interface dictionary |
| <code>getInterface</code> | Get the object for a named interface in an interface dictionary |
| <code>getInterfaces</code> | Get the object for a named interface in an interface dictionary |

See Also

`systemcomposer.interface.SignalElement`

Topics

“Define Interfaces”

Introduced in R2019a

systemcomposer.interface.SignalElement

Class that represents an element in the signal interface

Description

The `SignalElement` class represents a single element in the signal interface

Creation

Create a signal element.

```
addElement(interface,elementName)
```

Properties

Interface — Handle to the parent interface of the element

Interface object

Data Types: `systemcomposer.interface.SignalInterface`

Name — Element name

string

Dimensions — Dimensions of the element

array of positive integers

Type — Data type of the element

string

Complexity — complexity of the element

'real' | 'complex'

Units — Units of the element

string

Minimum — Minimum value for the element

double

Maximum — Maximum value for the element

double

Description — Description text for the element

string

Object Functions

destroy Remove and destroy a model element

See Also

addInterface

Topics

“Define Interfaces”

Introduced in R2019a

systemcomposer.interface.SignalInterface

Class that represents the structure of the signal interface

Description

The `SignalInterface` class represents the structure of the signal interface at a given port

Creation

Create an interface.

```
interface = addInterface(architecture, name)
```

Properties

Dictionary — Handle to the parent dictionary of the interface

Interface dictionary object

Data Types: `systemcomposer.interface.Dictionary`

Name — Interface name

string

Elements — Elements in interface

array of interface elements

Object Functions

| | |
|----------------------------|---|
| <code>addElement</code> | Add a signal interface element |
| <code>removeElement</code> | Remove a signal interface element |
| <code>getElement</code> | Get the object a signal interface element |
| <code>destroy</code> | Remove and destroy a model element |

See Also

systemcomposer.interface.SignalInterface

Topics

“Define Interfaces”

Introduced in R2019a

systemcomposer.profile.Profile

Class that represents a profile

Description

The Profile class represents architecture profiles.

Creation

```
profiles = <architecture>.Profiles;
```

Properties

Name — Name of the profile

string

Data Types: char

Description — Description text for the profile

string

Data Types: char

Object Functions

addStereotype Add a stereotype to the profile

removeStereotype Remove a stereotype from a model element

See Also

systemcomposer.profile.Stereotype

Topics

“Define Profiles and Stereotypes”

Introduced in R2019a

systemcomposer.profile.Property

Class that represents a property

Description

The Property class represents properties in a stereotype.

Creation

`addProperty(stereotype,AttributeName,AttributeValue)`

Properties

Name — Name of the property

string

Data Types: char

Name — Property name

string

Data Types: char

Datatype — Property data type

valid data type string

Data Types: char

Dimensions — Dimensions of property

positive integer array

Data Types: char

Min — Minimum value

numeric value

Data Types: double

Max — Maximum value

numeric value

Data Types: double

Units — Property units

string

Data Types: char

Object Functions

destroy Remove and destroy a model element

See Also

systemcomposer.profile.Profile | systemcomposer.profile.Stereotype

Topics

“Define Profiles and Stereotypes”

Introduced in R2019a

systemcomposer.profile.Stereotype

Class that represents a stereotype

Description

The Stereotype class represents architecture stereotypes in a profile.

Creation

```
addStereotype(profile, name, type)
```

Properties

Name — Name of the stereotype

string

Data Types: char

Description — Description text for the stereotype

string

Data Types: char

Icon — Icon for the stereotype

string

Data Types: char

Parent — The stereotype from which this stereotype inherits its properties

stereotype

Data Types: systemcomposer.profile.Stereotype

AppliesTo — The element type to which this stereotype can be applied

stereotype

Data Types: systemcomposer.profile.Stereotype

Abstract — Whether the stereotype is abstract

true | false

If true then stereotype cannot be directly applied on model elements, but instead serves as a parent for other stereotypes.

Properties — Array of property definitions owned or inherited by this stereotype

stereotype

Data Types: systemcomposer.profile.Stereotype

Object Functions

| | |
|----------------|-------------------------------------|
| addProperty | Add a property to a stereotype |
| removeProperty | Remove a property from a stereotype |
| reparent | Move stereotype |

See Also

systemcomposer.profile.Stereotype

Topics

“Define Profiles and Stereotypes”

Introduced in R2019a

