SENSTAR™

Symphony Hardware Interface SDK

7.x

Developer Guide

Contents

Introduction

The Symphony Hardware Interface SDK is a collection of .NET interfaces that can integrate hardware systems (for example, access control systems, I/O devices, and intrusion detection systems) with the Symphony Server.

Using the Symphony Hardware Interface SDK to integrate a hardware system with the Symphony Server enables the following functionality:

- System configuration
- Device status
- Alarms
- Commands

Getting started

1. Create a new .NET assembly DLL that includes an implementation of the `IHardwareSystem` and `IHardwareSystemFactory` interfaces in `Core.Interface.HardwareSystemFactory.dll`.
2. Put the new `Core.Interface.HardwareSystemFactory.dll` in the Symphony Server installation folder.

The Symphony Server automatically loads the DLL and the functionality is available on the Integrations page in the Symphony Server configuration interface.

Related reference

Basic system initialization flow on page 20

Sample

The `HardwareSystemSample` project is a sample of a hardware system integration.

The sample project consists of an implementation of the `IHardwareSystem` interface. The sample project can help you become familiar with the interface and you can use it as a template to for a new hardware system project.

Interfaces

IHardwareSystem

This interface defines the properties and methods that the Hardware System needs to implement to fully integrate with the Symphony Server.

Namespace

`Core.Interface.HardwarePack.System`

Properties

Name	Type	Description
`Id`	`string`	• The unique identifier of the system instance • Set by the IHardwareSytemFactory from Symphony Server
`Type`	`HardwareSystemType`	• The hardware system type
`Status`	`HardwareSystemStatus`	• The current status of the system
`Nodes`	`IEnumarable<HardwareNode>`	• Nodes tree that exists in the system • Thread safety: best practice to synchronize access to this property between the hardware system and the Symphony Server • Write to nodes when the status is `not Connected` and do not write to nodes when the status is `Connected`

Events

Name	Type	Description
`EventsReceived`	`HardwareEventReceivedHandler`	• Notification that an event has occurred (node and system level)
`StateChanged`	`HardwareStateChangedHandler`	• Notification that a state of a node has changed
`StatusChanged`	`HardwareStatusChangedHandler`	• Notification that the status of the system has changed

Methods

Name	Description	Parameters	Returns
`Connect`	• The Symphony Server calls `Connect` after creating the hardware system instance	NA	NA
`GetState`	• Returns the state of a node	`string nodeId` • The node ID to get the state from	`HardwareState` The state of the node requested

`Send Command`	• Sends the command	`string nodeId`
		• The node to send the command to
		`string commandId`
		• The command to send
			NA

Extension methods

Name	Description	Parameters	Returns
`GetAllNodes`	• Returns all nodes in the system as a flat list	NA	`IEnumerable<HardwareNode>`
`GetUniqueNodeTypes`	• Returns all distinct node types available in the system	NA	`IEnumerable<HardwareNodeType>`

IHardwareSystemFactory

This interface defines the properties and methods that the Hardware System Factory needs to implement to fully integrate with the Symphony Server.

Namespace

`Core.Interface.HardwarePack.System`

Properties

Name	Type	Description
`Type`	`HardwareSystemType`	• The hardware system type that this factory creates

Methods

Name	Description	Parameters	Returns

`Create`	• Creates a new hardware system instance	`string systemId`
		• The unique ID that the Symphony Server allocates to the system
		• The `IHardwareSystem on page 4.Id` property for the created system instance has this value
		`string settingsXml`
		• An XML string that holds the configuration settings of the system
		• For more information on configuration, see HardwareSystemConfiguration on page 8
		`IStatePersister Data`
		• An instance of IStatePersister that can save data to and load data from the Symphony Database
			NA

IStatePersister

This interface defines the methods that a hardware system can use to persist data in the Symphony Database. The data is saved even after the system is disposed, and can be loaded when the system is initialized.

Namespace

`Core.Interface.HardwarePack.Data`

Methods

Name	Description	Parameters	Returns
`Load`	• Load the data that was saved in the Symphony Database	NA	string
`Save`	• Save data in Symphony Database • Data links to the system ID.	`string data` • The data to be saved in the Symphony Database	NA

Classes

HardwareSystemType

This class represent the data of a system type that is used for enumerating system types and defining the configuration fields. This class is a property of IHardwareSystem on page 4 and IHardwareSystemFactory on page 5 interfaces.

Namespace

```
Core.Interface.HardwarePack.System
```

Properties

Name	Type	Description
`Id`	`string`	• The identifier of the hardware system type
`Name`	`string`	• The name of the hardware system type • This name appears on the Integrations page in the Symphony Server configuration interface
`Configuration`	`HardwareSystemConfiguration on page 8`	• The configuration details used to generate user interfaces for editing the settings XML • Set to null to use a basic text box to edit settings XML

HardwareSystemConfiguration

This class defines the configuration details used to generate the user interfaces for editing the settings XML of a hardware system.

Namespace

```
Core.Interface.HardwarePack.System
```

Properties

Name	Type	Description
`TemplateXml`	`string`	• A settings XML with no values or with default values

Name	Type	Description
Fields	HardwareSystemConfigurationField on page 9[]	• The definitions of the fields that are included in the configuration
Details	string[]	• An array of field IDs that are concatenated to generate summary details

HardwareSystemConfigurationField

This class defines the configuration fields used in HardwareSystemConfiguration on page 8.

Namespace

Core.Interface.HardwarePack.System

Properties

Name	Type	Description
Id	string	• The identifier for the field
Selector	string	• The selector to determine the location of field's XML element in the settingsXml • See https://api.jquery.com/category/selectors/

Type	string	• The field data type
		• All optional data types are const members of this class
		• Includes:
		  • TypeString
		  • TypeNumber
		  • TypeBoolean
		  • TypePassword
		  • TypeIPAddress

DefaultValue	string	• The field default value
MinimumNumber	int	• The minimum number using TypeNumber • Null if not used
MaximumNumber	int	• The maximum number using TypeNumber • Null if not used
Label	string	• The displayed label of the field

Name	Type	Description
Row	int	• The field's row in the configuration UI
Column	int	• The field's column in the configuration UI
IsRequired	bool	• Is the field required

HardwareNode

This class represents a hardware node in the system. A hardware node can represent a device, controller, input/output point, sensor, etc. The hardware node can have events triggered on, report its status, and send commands.

Namespace

Core.Interface.HardwarePack.Node

Properties

Name	Type	Description
Id	string	• The unique identifier of the node • Used in all node related operations (for example, send command, trigger event)
Name	string	• The display name of the node
Type	HardwareNodeType on page 11	• The type of the node • Defines the events/ commands/states it supports
Parent	HardwareNode	• The parent node in the nodes tree
Children	List<HardwareNode>	• The list of child nodes in the nodes tree
LocationRangeDescription	LRangeDescription on page 12	• The location description of the node • Only for nodes that can support location based events • Null if not used

Name	Type	Description
System	IHardwareSystem	• The system that the node is part of

Methods

Name	Description	Parameters	Returns
AddChild	• Adds a child node • Updates Parent property of the child and Children property of the current node	HardwareNode child	none

HardwareNodeType

This class represents a Hardware Node type. It defines the states/events/commands that the nodes from this type supports.

Namespace

Core.Interface.HardwarePack.Node

Properties

Name	Type	Description
Id	string	• The unique identifier of the Node Type
Name	string	• The display name of the type
AvailableEventTypes	List<HardwareEventType on page 13>	• A list of the events that the node type supports
AvailableCommands	List<HardwareCommand on page 12>	• A list of the commands that the node type supports
AvailableStates	List<HardwareState on page 13>	• A list of the states that the node type supports
AccessGrantedEventTypes	List<HardwareEventType on page 13>	• Only for Access Control type of nodes • A list of the events that the node type supports that represent Access Granted events • Null if not used

Name	Type	Description
AccessDeniedEventTypes	List<HardwareEventType on page 13>	• Only for Access Control type of nodes • A list of the events that the node type supports that represent Access Denied events • Null if not used

LRangeDescription

This class defines properties and methods that the hardware system needs to implement to fully integrate with the Symphony Server.

Namespace

Core.Interface.HardwarePack.Node

Properties

Name	Type	Description
Start	uint	• The start of the range
End	uint	• The end of the range

HardwareCommand

This class represents the hardware command data.

Namespace

Core.Interface.HardwarePack.Data

Properties

Name	Type	Description
Id	string	• The unique identifier of the command • Used when sending commands)
Name	string	• The display name of the command

Name	Type	Description
RequiredStates	IEnumarable<HardwareState on page 13>	• If this collection is not empty, the command is only available when node state is one of these states. • If this collection is empty, the command is always available regardless of node state.

HardwareEventType

This class represents the hardware event type data.

Namespace

Core.Interface.HardwarePack.Data

Properties

Name	Type	Description
Id	string	• A unique Identifier of the event
Name	string	• The display name of the event
Category	HardwareEventTypeCategory on page 17	• The category of the event

HardwareState

This class represents the hardware state data.

Namespace

Core.Interface.HardwarePack.Data

Properties

Name	Type	Description
Id	string	• A unique Identifier of the state
Name	string	• The display name of the state
Type	HardwareStateType on page 17	• The type of the state
AlarmLocationDetails	IEnumerable<LocationDetails on page 14>	• A collection of locations details where there are alarms (recent non active, or currently active)

LocationDetails

This class represents a location where there is currently an alarm (recently inactive or currently active).

Namespace

`Core.Interface.HardwarePack.Data`

Properties

Name	Type	
`DistancFromStartPoint`	`float?`	• Distance from the start of the range • Null if not available
`Latitude`	`float?`	• Absolute latitudinal position • Null if not available
`Longitude`	`float?`	• Absolute longitudinal position • Null if not available
`Altitude`	`float?`	• Absolute altitude • Null if not available
`IsActive`	`bool`	• Is the alarm currently active

HardwareEventArgs

This class defines the arguments that are sent on a hardware system event when triggered.

Namespace

`Core.Interface.HardwarePack.Event`

Properties

Name	Type	Description
`NodeId`	`string`	• Node identifier the event is triggered on
`NodeName`	`string`	• Node name the event is triggered on
`UserId`	`long`	• The id of the user that triggered the event • Null if not used
`UserDescription`	`string`	• The description of the user that triggered the event • Null if not used

Name	Type	Description
UserImageName	string	• The name of the user image that triggered the event • Null if not used
UserImageData	byte[]	• The user image that triggered the event • Null if not used
Location	LocationDetails on page 14	• The location of the event • Null if not used
SystemId	string	• The system identifier the event is triggered from
UtcTime	DateTime	• The time (in UTC) when the event occurs

HardwareStateEventArgs

This class defines the arguments that are sent on a hardware system state change when a node state changes.

Namespace

```
Core.Interface.HardwarePack.Event
```

Properties

Name	Type	Description
NodeId	string	• The identifier of the node for which the state changes
NodeName	string	• The name of the node for which the state changes
State	HardwareState on page 13	• The new state of the node
SystemId	string	• The identifier of the hardware system to which the node is a part
UtcTime	datetime	• The time (in UTC) when the state changes

HardwareStatusEventArgs

This class defines the arguments that are sent when the status of the hardware system changes.

Namespace

```
Core.Interface.HardwarePack.Event
```

Properties

Name	Type	Description
`Status`	`HardwareSystemStatus` on page 17	• The new status of the hardware system
`SystemId`	`string`	• The hardware system identifier
`UtcTime`	`datetime`	• The time (in UTC) when the status changes

Enumerations

HardwareSystemStatus

This enumeration defines the possible statuses of the hardware system.

Namespace

`Core.Interface.HardwarePack.System`

Values

Name	Description
`Error`	• The hardware system is in an unrecoverable error state • The Symphony Server disposes and recreates the system
`Initialized`	• The hardware system is initialized and waiting for a connection request
`Connecting`	• The hardware system is in the process of connecting
`Connected`	• The hardware system is connected

HardwareStateType

This enumeration defines the possible state types of the hardware node.

Namespace

`Core.Interface.HardwarePack.Data`

Values

Name	Description
`Normal`	• The state is the normal state
`Warning`	• The state is a warning state
`Alert`	• The state is an alert state

HardwareEventTypeCategory

This enumeration defines properties and methods that a hardware system needs to implement to fully integrate with the Symphony Server.

Namespace

`Core.Interface.HardwarePack.Data`

Values

Name	Description
`Default`	• The default category
`PIDS`	• The PIDS intrusion detection system category
`AccessGranted`	• The access granted category • Used for access control systems
`AccessDenied`	• The access denied category • Used for access control systems

Delegates

HardwareEventReceivedHandler

This delegate represents the method that handles an event received from a hardware system.

Namespace

`Core.Interface.HardwarePack.Event`

Parameters

Name	Type	Description
e	HardwareEventArgs	• An object that contains the arguments of the event

HardwareStateChangedHandler

This delegate represents the method that handles a node state change event received from a hardware system.

Namespace

`Core.Interface.HardwarePack.Event`

Parameters

Name	Type	Description
e	HardwareStateEventArgs	• An object that contains the arguments of the node state change event

HardwareStatusChangedHandler

This delegate represents the method that handles a system status event received from a hardware system.

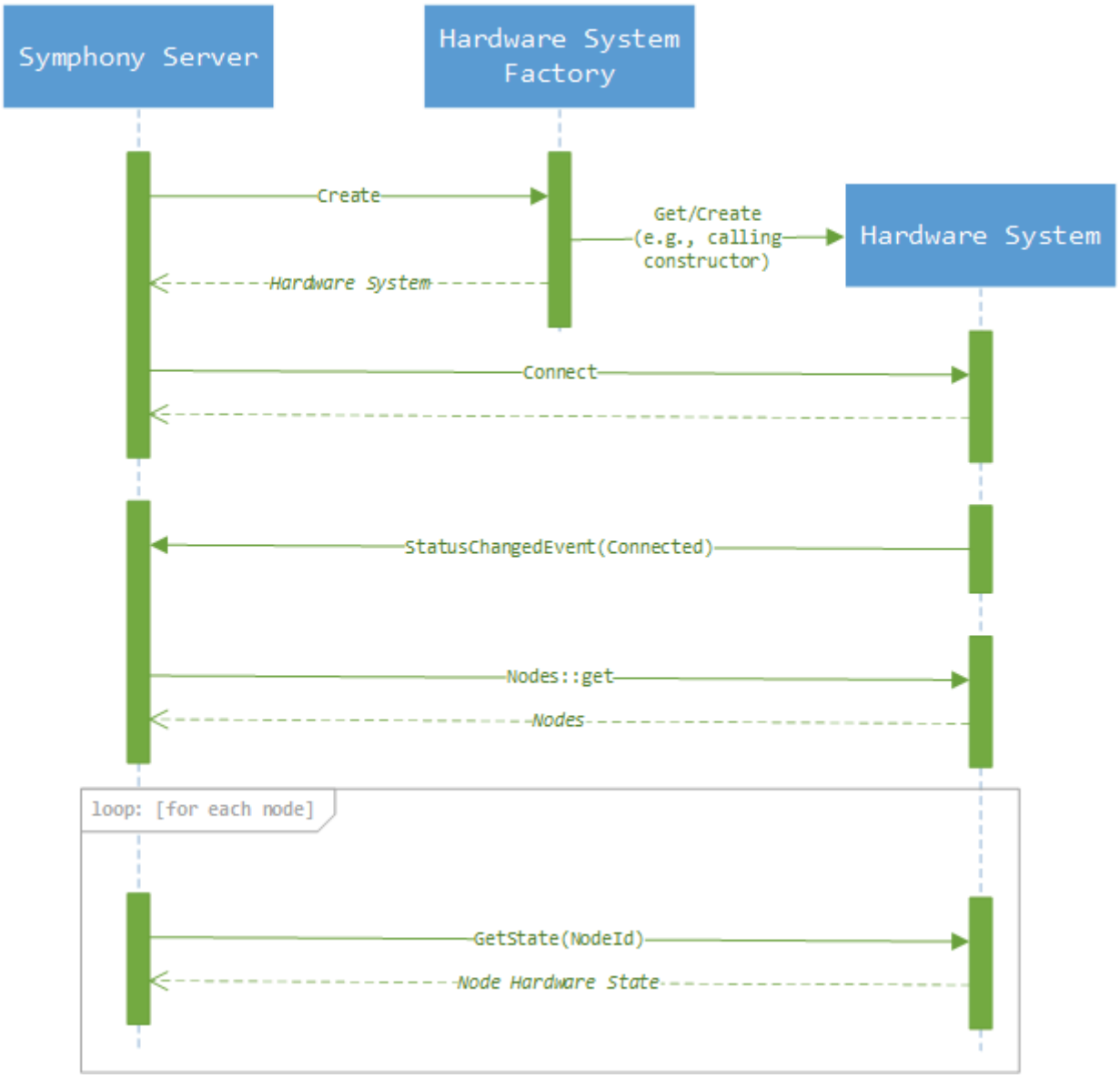
Namespace

`Core.Interface.HardwarePack.Event`

Parameters

Name	Type	Description
e	HardwareStatusEventArgs	• An object that contains the arguments of the system status event

Basic system initialization flow



Related tasks

Getting started on page 3

Trobleshooting

Log files

The log files for the library can be found in the configured log folder. The file format for the log files is `is-yymmdd_n.txt`.

Legal information

This material is for informational purposes only. Senstar makes no warranties, express, implied or statutory, as to the information in this document.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Senstar Corporation

Senstar may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Senstar, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Senstar and the Senstar logo are registered trademarks of Senstar Corporation.

All other trademarks are the property of their respective owners.