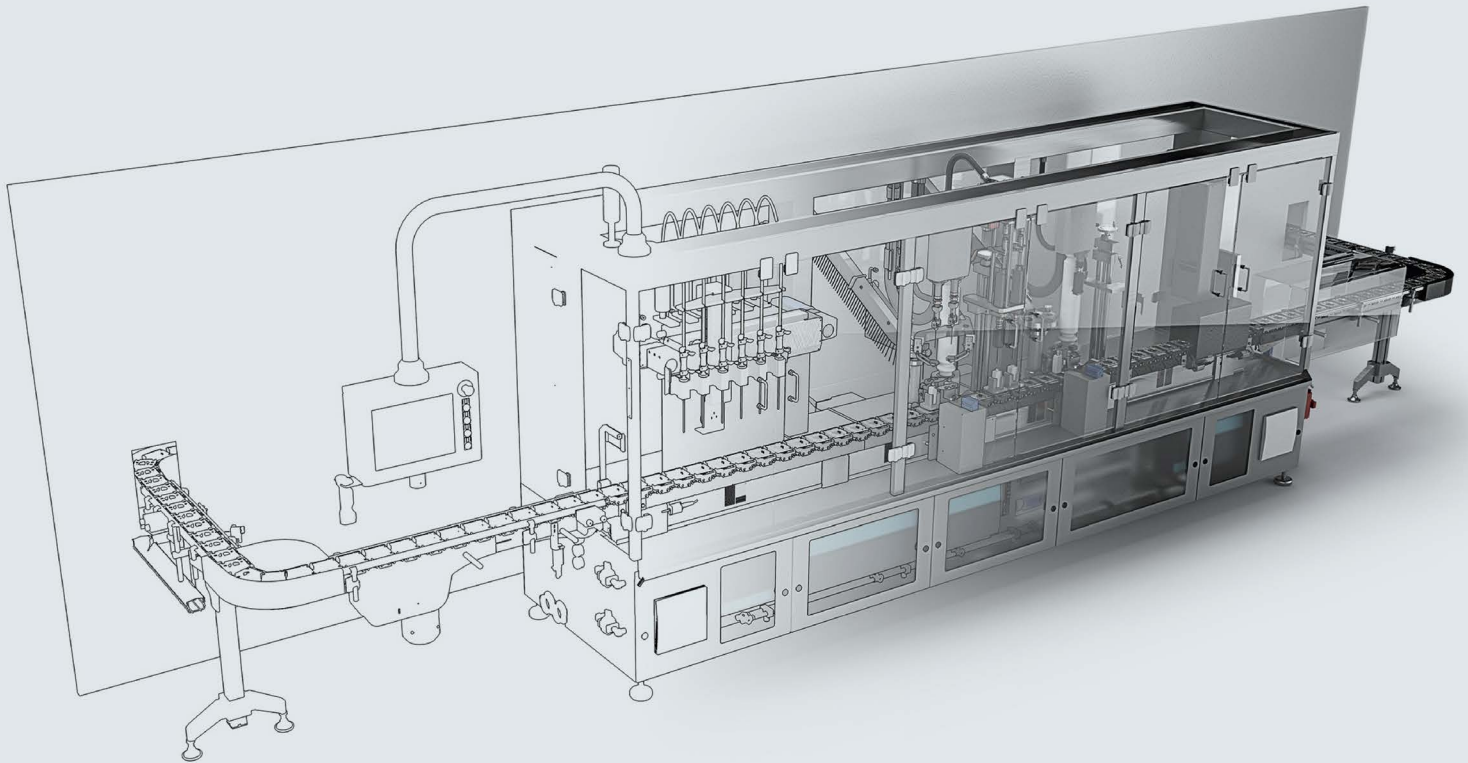


**SIEMENS**



# SIMATIC

S7-1500

S7-PLCSIM Advanced

Function manual

Edition

09/2016

[siemens.com](http://siemens.com)

# SIEMENS

## SIMATIC

### S7-1500 S7-PLCSIM Advanced

#### Function Manual

#### Preface

---

#### Guide

---

1

#### Product overview

---

2

#### Installing

---

3

#### Communication paths

---

4

#### Simulate CPU

---

5

#### Virtual time response

---

6

#### User interfaces (API)

---

7

#### Restrictions

---

8

#### List of abbreviations




---

A

## Legal information

### Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 <b>DANGER</b>
indicates that death or severe personal injury <b>will</b> result if proper precautions are not taken.
 <b>WARNING</b>
indicates that death or severe personal injury <b>may</b> result if proper precautions are not taken.
 <b>CAUTION</b>
indicates that minor personal injury can result if proper precautions are not taken.
<b>NOTICE</b>
indicates that property damage can result if proper precautions are not taken.


If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

### Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

### Proper use of Siemens products

Note the following:

 <b>WARNING</b>
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

### Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

### Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

# Preface

## Purpose of the documentation

This function manual describes the simulation software, S7-PLCSIM Advanced V1.0. It enables you to test your CPU programs on a simulated, virtual S7-1500 controller.

You can obtain updates to the function manual as downloads on the Internet (<https://support.industry.siemens.com/cs/us/en/view/109739154>).

## Scope

This function manual is valid for the following order versions

- 6ES7823-1FA00-0YA5 - S7-PLCSIM Advanced V1.0 DVD
- 6ES7823-1FE00-0YA5 - S7-PLCSIM Advanced V1.0 Download

The articles each contain one license for one instance.

## Basic knowledge required

The system may only be used by qualified staff.

The following knowledge is required:

- Industrial Automation and Automation Technology
- Programming with STEP 7 (TIA Portal)
- SIMATIC CPUs and CPU programming
- PC-based automation using S7-1500 and WinCC Runtime Advanced
- Development of software in C++ and C#
- PC technology
- Windows operating system

## Conventions

Conventions STEP 7: In this documentation, "STEP 7" is used as a synonym for all versions of the configuration and programming software "STEP 7 (TIA Portal)".

We also use abbreviate SIMATIC S7-PLCSIM Advanced V1.0 as "PLCSIM Advanced".

Please also observe notes marked as follows:

---

### Note

A note contains important information on the product described in the documentation, on the handling of the product or on the section of the documentation to which particular attention should be paid.

---

## Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit (<http://www.siemens.com/industrialsecurity>).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (<http://www.siemens.com/industrialsecurity>).

# Table of contents

	<b>Preface</b> .....	<b>4</b>
<b>1</b>	<b>Guide</b> .....	<b>22</b>
1.1	Documentation Guide S7 PLCSIM Advanced .....	22
1.2	S7 PLCSIM products .....	24
<b>2</b>	<b>Product overview</b> .....	<b>25</b>
2.1	What is PLCSIM Advanced?.....	25
2.2	Security for PLCSIM Advanced .....	26
<b>2.3</b>	<b>Simulations support</b> .....	<b>27</b>
2.4	Supported CPUs .....	28
2.5	Differences between a simulated and a real CPU .....	29
2.5.1	Restrictions for all supported CPUs .....	29
2.6	Simulating Motion Control.....	31
<b>3</b>	<b>Installing</b> .....	<b>32</b>
3.1	Introduction .....	32
3.1.1	System requirements .....	32
3.1.2	Licenses .....	33
3.1.3	Installation log .....	34
3.2	Installation medium .....	35
3.3	Installing PLCSIM Advanced .....	35
3.4	Changing PLCSIM Advanced .....	37
3.5	Repairing PLCSIM Advanced .....	38
3.6	Uninstalling PLCSIM Advanced .....	39
<b>4</b>	<b>Communication paths</b> .....	<b>40</b>
4.1	Local communication .....	41
4.2	Communication via TCP / IP.....	42
4.3	Enable distributed communication .....	43
<b>5</b>	<b>Simulate CPU</b> .....	<b>45</b>
5.1	Basic procedure for the simulation.....	45
5.2	User interface .....	46
5.2.1	S7 PLCSIM Advanced Symbol .....	46
5.2.2	S7 PLCSIM Advanced Control Panel .....	46
5.3	Download .....	50
5.4	MAC address of the instances .....	52
5.5	Simulate peripheral I/O .....	53
5.6	Simulate communication .....	54
5.6.1	Communication services that can be simulated .....	54
5.6.2	Communication between instances .....	55
5.7	Provide project data offline for simulation.....	56

<b>6</b>	<b>Virtual time response .....</b>	<b>58</b>
6.1	Speed up and slow down simulation .....	60
6.2	Stop simulation .....	61
6.3	Synchronize simulation partner .....	62
6.3.1	Synchronize simulation partner cycle-controlled .....	62
6.3.2	Synchronize simulation partner time-controlled.....	64
<b>7</b>	<b>User interfaces (API).....</b>	<b>65</b>
7.1	Introduction .....	65
7.1.1	Access to instances .....	67
7.1.2	User interfaces (API).....	68
7.1.3	Overview of user interfaces for native C++ .....	69
7.1.4	Overview of user interfaces for managed code .....	73
7.1.5	Overview of data types for native C++.....	76
7.1.6	Overview of data types for managed code .....	77
7.2	Initialize API .....	78
7.2.1	Native C++ .....	78
7.2.1.1	InitializeApi() .....	78
7.2.1.2	RuntimeApiEntry_Initialize .....	79
7.2.2	.NET (C#).....	80
7.2.2.1	Initialize .....	80
7.3	Shut down API .....	81
7.3.1	Native C++ .....	81
7.3.1.1	DestroyInterface() .....	82
7.3.1.2	RuntimeApiEntry_DestroyInterface.....	83
7.3.1.3	FreeApi() .....	84
7.3.1.4	ShutdownAndFreeApi().....	85
7.3.2	.NET (C#).....	86
7.3.2.1	Shut down API .....	86
7.4	Global functions (Native C++).....	86
7.5	API ISimulationRuntimeManager.....	90
7.5.1	Interfaces - Information and settings .....	90
7.5.2	Simulation Runtime instances .....	93
7.5.3	Remote connections .....	101
7.5.4	Events .....	106
7.5.4.1	OnConfigurationChanged .....	106
7.5.4.2	OnRuntimeManagerLost.....	110
7.6	API IInstances.....	113
7.6.1	Interfaces - Information and settings .....	113
7.6.2	Controller - Information and settings.....	119
7.6.3	Operating state .....	127
7.6.4	Tag list .....	135
7.6.5	I/O access .....	141
7.6.5.1	I/O access via address - Reading .....	141
7.6.5.2	I/O access via address - Writing .....	150
7.6.5.3	I/O access via tag name - Reading.....	158
7.6.5.4	I/O access via tag name - Writing.....	188

7.6.6	Settings for the virtual time .....	218
7.6.7	Cycle control .....	221
7.6.8	Events .....	227
7.6.8.1	OnOperatingStateChanged .....	228
7.6.8.2	OnEndOfCycle .....	232
7.6.8.3	OnConfigurationChanging .....	234
7.6.8.4	OnConfigurationChanged .....	237
7.6.8.5	OnLedChanged.....	239
7.7	API IRemoteRuntimeManager .....	242
7.7.1	Interfaces - Information and settings.....	242
7.7.2	Simulation Runtime instances.....	246
7.7.2.1	Simulation Runtime instances (remote).....	246
7.7.3	Events .....	254
7.7.3.1	OnConnectionLost .....	254
7.8	Data types .....	257
7.8.1	DLL import functions (Native C++).....	258
7.8.1.1	ApiEntry_Initialize .....	258
7.8.1.2	ApiEntry_DestroyInterface .....	258
7.8.2	Event callback functions (Native C++).....	259
7.8.2.1	EventCallback_VOID .....	259
7.8.2.2	EventCallback_II_SREC_ST .....	259
7.8.2.3	EventCallback_II_SREC_ST_SROS_SROS.....	260
7.8.2.4	EventCallback_II_SREC_ST_SRLT_SRLM .....	261
7.8.2.5	EventCallback_II_SREC_ST_INT64_UINT32 .....	262
7.8.2.6	EventCallback_IRRTM.....	263
7.8.2.7	EventCallback_SRCC_UINT32_UINT32_INT32.....	263
7.8.2.8	EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32.....	264
7.8.3	Delegate definitions (managed code).....	265
7.8.3.1	Delegate_Void.....	265
7.8.3.2	Delegate_II_EREC_DT .....	265
7.8.3.3	Delegate_II_EREC_DT_EOS_EOS.....	266
7.8.3.4	Delegate_II_EREC_DT_ELT_ELM.....	267
7.8.3.5	Delegate_II_EREC_DT_INT64_UINT32 .....	268
7.8.3.6	Delegate_IRRTM.....	269
7.8.3.7	Delegate_SRCC_UINT32_UINT32_INT32 .....	269
7.8.3.8	Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32 .....	270
7.8.4	Definitions and constants.....	271
7.8.5	Unions (Native C++) .....	272
7.8.5.1	UIP .....	272
7.8.5.2	UDataValue.....	272
7.8.6	Structures .....	274
7.8.6.1	SDataValue .....	274
7.8.6.2	SDataValueByAddress .....	276
7.8.6.3	SDataValueByName .....	276
7.8.6.4	SConnectionInfo.....	277
7.8.6.5	SInstanceInfo .....	278
7.8.6.6	SDimension.....	279
7.8.6.7	STagInfo.....	280
7.8.6.8	SIP .....	282
7.8.6.9	SIPSuite4 .....	283



7.8.7	Enumerations .....	284
7.8.7.1	ERuntimeErrorCode .....	284
7.8.7.2	EArea .....	286
7.8.7.3	EOperatingState .....	287
7.8.7.4	EOperatingMode .....	288
7.8.7.5	ECPUType .....	289
7.8.7.6	ECommunicationInterface .....	291
7.8.7.7	ELEDType .....	292
7.8.7.8	ELEDMode .....	293
7.8.7.9	EPrimitiveDataType .....	294
7.8.7.10	EDataType .....	296
7.8.7.11	ETagListDetails .....	299
7.8.7.12	ERuntimeConfigChanged .....	300
7.8.7.13	EInstanceConfigChanged .....	300
<b>8</b>	<b>Restrictions .....</b>	<b>302</b>
8.1	Overview .....	302
8.2	OPC UA server .....	302
8.3	Web server .....	304
8.4	Restrictions for communications services .....	305
8.5	Restrictions for instructions .....	306
8.6	Restrictions with Motion Control .....	307
8.6.1	Motion Control resources .....	307
8.6.2	Technology modules .....	308
8.7	Restrictions to local communication via Softbus .....	308
8.8	Restrictions of security with VMware vSphere Hypervisor (ESXi) .....	309
8.9	Error with overflow cyclical events .....	309
8.10	Deviating I/O values in the STEP 7 user program .....	310
8.11	Multiple simulations and possible collision of IP addresses .....	310
8.12	Lacking access to an IP address .....	310
8.13	Simulation in standby mode .....	310
8.14	Error installing the antivirus software from Kaspersky .....	311
<b>A</b>	<b>List of abbreviations .....</b>	<b>312</b>

## Tables

Table 3- 1	Minimum requirements for hardware and software .....	32
Table 5- 1	Assignment of the Ethernet interfaces, for example, a CPU 1518-4 PN/DP .....	52
Table 5- 2	Supported communication options.....	54
Table 7- 1	Overview of initializing and shutting down API - Native C++ .....	69
Table 7- 2	Overview of global functions - Native C++ .....	69
Table 7- 3	Overview of API ISimulationRuntimeManager functions - Native C++ .....	70
Table 7- 4	Overview of IInstances functions - Native C++ .....	70
Table 7- 5	Overview of IRemoteRuntimeManager functions - Native C++ .....	72
Table 7- 6	Overview of initializing and shutting down API - .NET (C#).....	73
Table 7- 7	Overview of ISimulationRuntimeManager functions - .NET (C#).....	73
Table 7- 8	Overview of IInstances functions - .NET (C#).....	74
Table 7- 9	Overview of IRemoteRuntimeManager functions - .NET (C#).....	75
Table 7- 10	Overview of data types - Native C++ .....	76
Table 7- 11	Overview of data types - .NET (C#) .....	77
Table 7- 12	InitializeApi() - Native C++ .....	78
Table 7- 13	RuntimeApiEntry_Initialize - Native C++ .....	79
Table 7- 14	Initialize - .NET (C#).....	80
Table 7- 15	DestroyInterface() - Native C++ .....	82
Table 7- 16	RuntimeApiEntry_DestroyInterface() - Native C++ .....	83
Table 7- 17	FreeApi() - Native C++ .....	84
Table 7- 18	ShutdownAndFreeApi() - Native C++ .....	85
Table 7- 19	GetNameOfAreaSection() - Native C++.....	86
Table 7- 20	GetNameOfCPUType() - Native C++.....	86
Table 7- 21	GetNameOfCommunicationInterface() - Native C++ .....	87
Table 7- 22	GetNameOfDataType() - Native C++.....	87
Table 7- 23	GetNameOfErrorCode() - Native C++.....	87
Table 7- 24	GetNameOfLEDMode() - Native C++ .....	87
Table 7- 25	GetNameOfLEDType() - Native C++ .....	88
Table 7- 26	GetNameOfOperatingMode() - Native C++ .....	88
Table 7- 27	GetNameOfErrorCode() - Native C++.....	88
Table 7- 28	GetNameOfOperatingState() - Native C++ .....	88
Table 7- 29	GetNameOfPrimitiveDataType() - Native C++.....	89
Table 7- 30	GetNameOfTagListDetails() - Native C++ .....	89
Table 7- 31	GetNameOfRuntimeConfigChanged() - Native C++.....	89
Table 7- 32	GetNameOfInstanceConfigChanged() - Native C++ .....	89

Table 7- 33	GetVersion() - Native C++ .....	90
Table 7- 34	Version { get; } - .NET (C#) .....	90
Table 7- 35	IsInitialized() - Native C++ .....	91
Table 7- 36	IsInitialized { get; } - .NET (C#) .....	91
Table 7- 37	IsRuntimeManagerAvailable() - Native C++ .....	91
Table 7- 38	IsRuntimeManagerAvailable { get; } - .NET (C#) .....	91
Table 7- 39	Shutdown() - Native C++ .....	92
Table 7- 40	Shutdown() - .NET (C#) .....	92
Table 7- 41	GetRegisteredInstancesCount() - Native C++ .....	93
Table 7- 42	GetRegisteredInstanceInfoAt() - Native C++ .....	93
Table 7- 43	RegisteredInstanceInfo { get; } - .NET (C#) .....	94
Table 7- 44	RegisterInstance() - Native C++ .....	95
Table 7- 45	RegisterInstance() - .NET (C#) .....	96
Table 7- 46	RegisterCustomInstance() - Native C++ .....	97
Table 7- 47	RegisterCustomInstance() - .NET (C#) .....	98
Table 7- 48	CreateInterface() - Native C++ .....	99
Table 7- 49	CreateInterface() - .NET (C#) .....	100
Table 7- 50	OpenPort() - Native C++ .....	101
Table 7- 51	OpenPort() - .NET (C#) .....	101
Table 7- 52	ClosePort() - Native C++ .....	102
Table 7- 53	ClosePort() - .NET (C#) .....	102
Table 7- 54	GetPort() - Native C++ .....	102
Table 7- 55	Port { get; } - .NET (C#) .....	102
Table 7- 56	GetRemoteConnectionsCount() - Native C++ .....	103
Table 7- 57	GetRemoteConnectionInfoAt()- Native C++ .....	103
Table 7- 58	RemoteConnectionInfo { get; } - .NET (C#) .....	103
Table 7- 59	RemoteConnect() - Native C++ .....	104
Table 7- 60	RemoteConnect() - .NET (C#) .....	105
Table 7- 61	Events for the ISimulationRuntimeManager interface .....	106
Table 7- 62	OnConfigurationChanged - .NET (C#) .....	106
Table 7- 63	RegisterOnConfigurationChangedCallback() - Native C++ .....	107
Table 7- 64	RegisterOnConfigurationChangedEvent() - Native C++ .....	107
Table 7- 65	RegisterOnConfigurationChangedEvent() - .NET (C#) .....	107
Table 7- 66	UnregisterOnConfigurationChangedCallback() - Native C++ .....	108
Table 7- 67	UnregisterOnConfigurationChangedEvent() - Native C++ .....	108
Table 7- 68	UnregisterOnConfigurationChangedEvent() - .NET (C#) .....	108

Table 7- 69	WaitForOnConfigurationChangedEvent() - Native C++ .....	109
Table 7- 70	WaitForOnConfigurationChangedEvent - .NET (C#) .....	109
Table 7- 71	OnRuntimeManagerLost - .NET (C#) .....	110
Table 7- 72	RegisterOnRuntimeManagerLostCallback() - Native C++ .....	110
Table 7- 73	RegisterOnRuntimeManagerLostEvent() - Native C++ .....	111
Table 7- 74	RegisterOnRuntimeManagerLostEvent() - .NET (C#) .....	111
Table 7- 75	UnregisterOnRuntimeManagerLostCallback() - Native C++ .....	111
Table 7- 76	UnregisterOnRuntimeManagerLostEvent() - Native C++ .....	112
Table 7- 77	UnregisterOnRuntimeManagerLostEvent() - .NET (C#) .....	112
Table 7- 78	WaitForOnRuntimeManagerLostEvent() - Native C++ .....	112
Table 7- 79	WaitForOnRuntimeManagerLostEvent() - .NET (C#) .....	112
Table 7- 80	Dispose() - .NET (C#) .....	113
Table 7- 81	GetID() - Native C++ .....	113
Table 7- 82	ID { get; } - .NET (C#) .....	113
Table 7- 83	GetName() - Native C++ .....	114
Table 7- 84	Name { get; } - .NET (C#) .....	114
Table 7- 85	GetCPUType() - Native C++ .....	115
Table 7- 86	SetCPUType() - Native C++ .....	115
Table 7- 87	CPUType { get; set; } - .NET (C#) .....	115
Table 7- 88	GetCommunicationInterface() - Native C++ .....	116
Table 7- 89	SetCommunicationInterface() - Native C++ .....	116
Table 7- 90	CommunicationInterface { get; set; } - .NET (C#) .....	117
Table 7- 91	GetInfo() - Native C++ .....	117
Table 7- 92	Info { get; } - .NET (C#) .....	117
Table 7- 93	UnregisterInstance() - Native C++ .....	118
Table 7- 94	UnregisterInstance() - .NET (C#) .....	118
Table 7- 95	GetControllerName() - Native C++ .....	119
Table 7- 96	ControllerName { get; } - .NET (C#) .....	119
Table 7- 97	GetControllerShortDesignation() - Native C++ .....	120
Table 7- 98	ControllerShortDesignation { get; } - .NET (C#) .....	120
Table 7- 99	GetControllerIPCount() - Native C++ .....	120
Table 7- 100	GetControllerIP() - Native C++ .....	121
Table 7- 101	ControllerIP { get; } - .NET (C#) .....	121
Table 7- 102	GetControllerIPSuite4() Native C++ .....	121
Table 7- 103	ControllerIPSuite4 { get; } - .NET (#) .....	121
Table 7- 104	SetIPSuite() - Native C++ .....	122

Table 7- 105	SetIPSuite() - .NET (C#) .....	122
Table 7- 106	GetStoragePath() - Native C++.....	123
Table 7- 107	SetStoragePath() - Native C++ .....	124
Table 7- 108	StoragePath { get; set; } - .NET (C#) .....	124
Table 7- 109	ArchiveStorage() - Native C++ .....	125
Table 7- 110	ArchiveStorage() - .NET (C#).....	125
Table 7- 111	RetrieveStorage() - Native C++ .....	126
Table 7- 112	RetrieveStorage() - .NET (C#).....	126
Table 7- 113	GetOperatingState() - Native C++ .....	127
Table 7- 114	OperatingState { get; } - .NET (C#).....	128
Table 7- 115	PowerOn() - Native C++ .....	129
Table 7- 116	PowerOn() - .NET (C#) .....	130
Table 7- 117	PowerOff() - Native C++ .....	131
Table 7- 118	PowerOff() - .NET (C#) .....	131
Table 7- 119	MemoryReset() - Native C++ .....	132
Table 7- 120	MemoryReset() - .NET (C#).....	132
Table 7- 121	Run() - Native C++.....	133
Table 7- 122	Run() - .NET (C#).....	133
Table 7- 123	Stop() - Native C++ .....	134
Table 7- 124	Stop() - .NET (C#).....	134
Table 7- 125	UpdateTagList() - Native C++ .....	136
Table 7- 126	UpdateTagList() - .NET (C#).....	137
Table 7- 127	GetTagListStatus() - Native C++.....	138
Table 7- 128	GetTagListStatus() - .NET (C#) .....	138
Table 7- 129	GetTagInfoCount() - Native C++.....	138
Table 7- 130	GetTagInfos() - Native C++.....	139
Table 7- 131	TagInfos { get; } - .NET (C#) .....	139
Table 7- 132	CreateConfigurationFile() - Native C++ .....	140
Table 7- 133	CreateConfigurationFile() - .NET (C#) .....	140
Table 7- 134	InputArea { get; } MarkerArea { get; } OutputArea { get; } - .NET (C#) .....	141
Table 7- 135	GetAreaSize() - Native C++ .....	141
Table 7- 136	AreaSize { get; } - .NET (C#).....	141
Table 7- 137	ReadBit() - Native C++.....	142
Table 7- 138	ReadBit() - .NET (C#) .....	143
Table 7- 139	ReadByte() - Native C++.....	144
Table 7- 140	ReadByte() - .NET (C#) .....	145

Table 7- 141	ReadByte() - Native C++ .....	146
Table 7- 142	ReadBytes() - .NET (C#).....	147
Table 7- 143	ReadSignals() - Native C++ .....	148
Table 7- 144	ReadSignals() - .NET (C#).....	149
Table 7- 145	WriteBit() - Native C++ .....	150
Table 7- 146	WriteBit() - .NET (C#).....	151
Table 7- 147	WriteByte() - Native C++ .....	152
Table 7- 148	WriteByte() - .NET (C#).....	153
Table 7- 149	WriteBytes() - Native C++ .....	154
Table 7- 150	WriteBytes() - .NET (C#).....	155
Table 7- 151	WriteSignals() - Native C++ .....	156
Table 7- 152	WriteSignals() - .NET (C#).....	157
Table 7- 153	Read() - Native C++ .....	158
Table 7- 154	Read() - .NET (C#).....	159
Table 7- 155	ReadBool() - Native C++.....	160
Table 7- 156	ReadBool() - .NET (C#).....	161
Table 7- 157	ReadInt8() - Native C++ .....	162
Table 7- 158	ReadInt8() - .NET (C#).....	163
Table 7- 159	ReadInt16() - Native C++ .....	164
Table 7- 160	ReadInt16() - .NET (C#).....	165
Table 7- 161	ReadInt32() - Native C++ .....	166
Table 7- 162	ReadInt32() - .NET (C#).....	167
Table 7- 163	ReadInt64() - Native C++ .....	168
Table 7- 164	ReadInt64() - .NET (C#).....	169
Table 7- 165	ReadUInt8() - Native C++ .....	170
Table 7- 166	ReadUInt8() - .NET (C#) .....	171
Table 7- 167	ReadUInt16() - Native C++ .....	172
Table 7- 168	ReadUInt16() - .NET (C#) .....	173
Table 7- 169	ReadUInt32() - Native C++ .....	174
Table 7- 170	ReadUInt32() - .NET (C#) .....	175
Table 7- 171	ReadInt64() - Native C++ .....	176
Table 7- 172	ReadUInt64() - .NET (C#) .....	177
Table 7- 173	ReadFloat() - Native C++ .....	178
Table 7- 174	ReadFloat() - .NET (C#).....	179
Table 7- 175	ReadDouble() - Native C++.....	180
Table 7- 176	ReadDouble() - .NET (C#) .....	181

Table 7- 177	ReadChar() - Native C++ .....	182
Table 7- 178	ReadChar() - .NET (C#) .....	183
Table 7- 179	ReadWChar() - Native C++ .....	184
Table 7- 180	ReadWChar() - .NET (C#) .....	185
Table 7- 181	ReadSignals() - Native C++ .....	186
Table 7- 182	ReadSignals() - .NET (C#) .....	187
Table 7- 183	Write() - Native C++ .....	188
Table 7- 184	Write() - .NET (C#) .....	189
Table 7- 185	WriteBool() - Native C++ .....	190
Table 7- 186	WriteBool() - .NET (C#) .....	191
Table 7- 187	WriteInt8() - Native C++ .....	192
Table 7- 188	WriteInt8() - .NET (C#) .....	193
Table 7- 189	WriteInt16() - Native C++ .....	194
Table 7- 190	WriteInt16() - .NET (C#) .....	195
Table 7- 191	WriteInt32() - Native C++ .....	196
Table 7- 192	WriteInt32() - .NET (C#) .....	197
Table 7- 193	WriteInt64() - Native C++ .....	198
Table 7- 194	WriteInt64() - .NET (C#) .....	199
Table 7- 195	WriteUInt8() - Native C++ .....	200
Table 7- 196	WriteUInt8() - .NET (C#) .....	201
Table 7- 197	WriteUInt16() - Native C++ .....	202
Table 7- 198	WriteUInt16() - .NET (C#) .....	203
Table 7- 199	WriteUInt32() - Native C++ .....	204
Table 7- 200	WriteUInt32() - .NET (C#) .....	205
Table 7- 201	WriteUInt64() - Native C++ .....	206
Table 7- 202	WriteUInt64() - .NET (C#) .....	207
Table 7- 203	WriteFloat() - Native C++ .....	208
Table 7- 204	WriteFloat() - .NET (C#) .....	209
Table 7- 205	WriteDouble() - Native C++ .....	210
Table 7- 206	WriteDouble() - .NET (C#) .....	211
Table 7- 207	WriteChar() - Native C++ .....	212
Table 7- 208	WriteChar() - .NET (C#) .....	213
Table 7- 209	WriteWChar() - Native C++ .....	214
Table 7- 210	WriteWChar() - .NET (C#) .....	215
Table 7- 211	WriteSignals() - Native C++ .....	216
Table 7- 212	WriteSignals() - .NET (C#) .....	217

Table 7- 213	GetSystemTime() - Native C++	218
Table 7- 214	SetSystemTime() - Native C++	218
Table 7- 215	SystemTime { get; set; } - .NET (C#)	218
Table 7- 216	GetScaleFactor() - Native C++	219
Table 7- 217	SetScaleFactor() - Native C++	219
Table 7- 218	ScaleFactor { get; set; } - .NET (C#)	220
Table 7- 219	GetOperatingMode() - Native C++	221
Table 7- 220	SetOperatingMode() - Native C++	221
Table 7- 221	OperatingMode { get; set; } - .NET (C#)	221
Table 7- 222	SetAlwaysSendOnEndOfCycleEnabled() - Native C++	222
Table 7- 223	IsAlwaysSendOnEndOfCycleEnabled() - Native C++	222
Table 7- 224	IsAlwaysSendOnEndOfCycleEnabled { get; set; } - .NET (C#)	222
Table 7- 225	GetOverwrittenMinimalCycleTime_ns() - Native C++	223
Table 7- 226	SetOverwrittenMinimalCycleTime_ns() - Native C++	223
Table 7- 227	OverwrittenMinimalCycleTime_ns { get; set; } - .NET (C#)	224
Table 7- 228	RunNextCycle() - Native C++	225
Table 7- 229	RunNextCycle() - .NET (C#)	225
Table 7- 230	StartProcessing() - Native C++	226
Table 7- 231	StartProcessing() - .NET (C#)	226
Table 7- 232	Events for the IInstances interface	227
Table 7- 233	OnOperatingStateChanged - .NET (C#)	228
Table 7- 234	RegisterOnOperatingStateChangedCallback() - Native C++	228
Table 7- 235	RegisterOnOperatingStateChangedEvent() - Native C++	229
Table 7- 236	UnregisterOnOperatingStateChangedCallback() - Native C++	230
Table 7- 237	UnregisterOnOperatingStateChangedEvent() - Native C++	231
Table 7- 238	UnregisterOnOperatingStateChangedEvent() - .NET (C#)	231
Table 7- 239	WaitForOnOperatingStateChangedEvent() - Native C++	231
Table 7- 240	WaitForOnOperatingStateChangedEvent() - .NET (C#)	231
Table 7- 241	OnEndOfCycle - .NET (C#)	232
Table 7- 242	RegisterOnEndOfCycleCallback() - Native C++	232
Table 7- 243	RegisterOnEndOfCycleEvent() - Native C++	232
Table 7- 244	UnregisterOnEndOfCycleCallback() - Native C++	233
Table 7- 245	RegisterOnEndOfCycleEvent() - Native C++	233
Table 7- 246	UnregisterOnEndOfCycleEvent() - .NET (C#)	233
Table 7- 247	WaitForOnEndOfCycleEvent() - Native C++	234
Table 7- 248	WaitForOnEndOfCycleEvent() - .NET (C#)	234



Table 7- 249	OnConfigurationChanging - .NET (C#).....	234
Table 7- 250	RegisterOnConfigurationChangingCallback() - Native C++ .....	235
Table 7- 251	RegisterOnConfigurationChangingEvent() - Native C++ .....	235
Table 7- 252	UnregisterOnConfigurationChangingCallback() - Native C++ .....	235
Table 7- 253	UnregisterOnConfigurationChangingEvent() - Native C++ .....	236
Table 7- 254	UnregisterOnConfigurationChangingEvent() - .NET (C#).....	236
Table 7- 255	WaitForOnConfigurationChangingEvent() - Native C++ .....	236
Table 7- 256	WaitForOnConfigurationChangingEvent() - .NET (C#).....	236
Table 7- 257	OnConfigurationChanged - .NET (C#).....	237
Table 7- 258	RegisterOnConfigurationChangedCallback() - Native C++ .....	237
Table 7- 259	RegisterOnConfigurationChangedEvent() - Native C++ .....	237
Table 7- 260	UnregisterOnConfigurationChangedCallback() - Native C++ .....	238
Table 7- 261	UnregisterOnConfigurationChangedEvent() - Native C++ .....	238
Table 7- 262	UnregisterOnConfigurationChangedEvent() - .NET (C#) .....	238
Table 7- 263	WaitForOnConfigurationChangedEvent() - Native C++ .....	239
Table 7- 264	WaitForOnConfigurationChangedEvent() - .NET (C#) .....	239
Table 7- 265	OnLedChanged - .NET (C#) .....	239
Table 7- 266	RegisterOnLedChangedCallback() - Native C++ .....	240
Table 7- 267	RegisterOnLedChangedEvent() - Native C++ .....	240
Table 7- 268	UnregisterOnLedChangedCallback() - Native C++ .....	240
Table 7- 269	UnregisterOnLedChangedEvent() - Native C++ .....	241
Table 7- 270	UnregisterOnLedChangedEvent() - .NET (C#).....	241
Table 7- 271	WaitForOnLedChangedEvent() - Native C++ .....	241
Table 7- 272	WaitForOnLedChangedEvent() - .NET (C#).....	241
Table 7- 273	Dispose() - .NET (C#) .....	242
Table 7- 274	GetVersion() - Native C++ .....	242
Table 7- 275	Version { get; } - .NET (C#) .....	242
Table 7- 276	GetIP() - Native C++ .....	243
Table 7- 277	IP { get; } - .NET (C#) .....	243
Table 7- 278	GetPort() - Native C++ .....	243
Table 7- 279	Port { get; } - .NET (C#).....	243
Table 7- 280	GetRemoteComputerName() - Native C++ .....	244
Table 7- 281	RemoteComputerName { get; } - .NET (C#).....	244
Table 7- 282	Disconnect() - Native C++ .....	245
Table 7- 283	Disconnect() - .NET (C#) .....	245
Table 7- 284	GetRegisteredInstancesCount() - Native C++ .....	246

Table 7- 285	GetRegisteredInstanceInfoAt() - Native C++ .....	246
Table 7- 286	RegisterInstanceInfo { get; } - .NET (C#) .....	247
Table 7- 287	RegisterInstance() - Native C++ .....	248
Table 7- 288	RegisterInstance() - .NET (C#) .....	249
Table 7- 289	RegisterCustomInstance() - Native C++ .....	250
Table 7- 290	RegisterCustomInstance() - .NET (C#) .....	251
Table 7- 291	CreateInterface() - Native C++ .....	252
Table 7- 292	CreateInterface() - .NET (C#) .....	253
Table 7- 293	OnConnectionLost - .NET (C#) .....	254
Table 7- 294	RegisterOnConnectionLostCallback() - Native C++ .....	254
Table 7- 295	RegisterOnConnectionLostEvent() - Native C++ .....	255
Table 7- 296	RegisterOnConnectionLostEvent() - .NET (C#) .....	255
Table 7- 297	UnregisterOnConnectionLostCallback() - Native C++ .....	255
Table 7- 298	UnregisterOnConnectionLostEvent() - Native C++ .....	256
Table 7- 299	UnregisterOnConnectionLostEvent() - .NET (C#) .....	256
Table 7- 300	WaitForOnConnectionLostEvent() - Native C++ .....	256
Table 7- 301	WaitForOnConnectionLostEvent() - .NET (C#) .....	256
Table 7- 302	ApiEntry_Initialize - Native C++ .....	258
Table 7- 303	ApiEntry_DestroyInterface - Native C++ .....	258
Table 7- 304	EventCallback_VOID - Native C++ .....	259
Table 7- 305	EventCallback_II_SREC_ST - Native C++ .....	259
Table 7- 306	EventCallback_II_SREC_ST_SROS_SROS - Native C++ .....	260
Table 7- 307	EventCallback_II_SREC_ST_SRLT_SRLM - Native C++ .....	261
Table 7- 308	EventCallback_II_SREC_ST_INT64_UINT32 - Native C++ .....	262
Table 7- 309	EventCallback_IRRTM - Native C++ .....	263
Table 7- 310	EventCallback_SRCC_UINT32_UINT32_INT32 - Native C++ .....	263
Table 7- 311	EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32 - Native C++ .....	264
Table 7- 312	Delegate_Void - .NET (C#) .....	265
Table 7- 313	Delegate_II_EREC_DT - .NET (C#) .....	265
Table 7- 314	Delegate_II_EREC_DT_EOS_EOS - .NET (C#) .....	266
Table 7- 315	Delegate_II_EREC_DT_ELT_ELM - .NET (C#) .....	267
Table 7- 316	Delegate_II_EREC_DT_INT64_UINT32 - .NET (C#) .....	268
Table 7- 317	Delegate_IRRTM - .NET (C#) .....	269
Table 7- 318	Delegate_SRCC_UINT32_UINT32_INT32 - .NET (C#) .....	269
Table 7- 319	Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32 - .NET (C#) .....	270
Table 7- 320	Definitions and constants - Native C++ .....	271

Table 7- 321	Definitions and constants - .NET (C#)	271
Table 7- 322	UIP - Native C++	272
Table 7- 323	UDataValue - Native C++	272
Table 7- 324	SDataValue - Native C++	274
Table 7- 325	SDataValue - .NET (C#)	274
Table 7- 326	SDataValueByAddress - Native C++	276
Table 7- 327	SDataValueByAddress - .NET (C#)	276
Table 7- 328	SDataValueByName - Native C++	276
Table 7- 329	SDataValueByName - .NET (C#)	276
Table 7- 330	SConnectionInfo - Native C++	277
Table 7- 331	SConnectionInfo - .NET (C#)	277
Table 7- 332	SInstanceInfo - Native C++	278
Table 7- 333	SInstanceInfo - .NET (C#)	278
Table 7- 334	SDimension - Native C++	279
Table 7- 335	SDimension - .NET (C#)	279
Table 7- 336	STagInfo - Native C++	280
Table 7- 337	STagInfo - .NET (C#)	281
Table 7- 338	SIP - .NET (C#)	282
Table 7- 339	SIPSuite4 - Native C++	283
Table 7- 340	SIPSuite4 - .NET (C#)	283
Table 7- 341	ERuntimeErrorCode - Native C++	284
Table 7- 342	ERuntimeErrorCode - .NET (C#)	285
Table 7- 343	EArea - Native C++	286
Table 7- 344	EArea - .NET (C#)	286
Table 7- 345	EOperatingState - Native C++	287
Table 7- 346	EOperatingState - .NET (C#)	287
Table 7- 347	EOperatingMode - Native C++	288
Table 7- 348	EOperatingMode - .NET (C#)	288
Table 7- 349	ECPUType - Native C++	289
Table 7- 350	ECPUType - .NET (C#)	290
Table 7- 351	ECommunicationInterface - Native C++	291
Table 7- 352	ECommunicationInterface - .NET (C#)	291
Table 7- 353	ELEDType - Native C++	292
Table 7- 354	ELEDType - .NET (C#)	292
Table 7- 355	ELEDMode - Native C++	293
Table 7- 356	ELEDMode - .NET (C#)	293

Table 7- 357	EPrimitiveDataType - Native C++ .....	294
Table 7- 358	EPrimitiveDataType - .NET (C#).....	294
Table 7- 359	Compatible primitive data types - Reading .....	295
Table 7- 360	Compatible primitive data types - Write .....	295
Table 7- 361	EDataType - Native C++ .....	297
Table 7- 362	EDataType - .NET (C#).....	298
Table 7- 363	ETagListDetails - Native C++ .....	299
Table 7- 364	ETagListDetails - .NET (C#).....	299
Table 7- 365	ERuntimeConfigChanged - Native C++ .....	300
Table 7- 366	ERuntimeConfigChanged - .NET (C#).....	300
Table 7- 367	EInstanceConfigChanged - Native C++ .....	300
Table 7- 368	EInstanceConfigChanged - .NET (C#).....	300
Table 8- 1	Instructions not supported.....	306
Table 8- 2	CPUs with limited Motion Control resources.....	307

## Figures

Figure 2-1	Enable simulation capability.....	27
Figure 4-1	Local communication via Softbus .....	41
Figure 4-2	Local communication via TCP/IP .....	42
Figure 4-3	Distributed communication via Ethernet .....	42
Figure 4-4	Distributed communication via network adapters .....	43
Figure 4-5	Activate PLCSIM Virtual Switch .....	44
Figure 4-6	Accessible devices on the Virtual Ethernet Adapter .....	44
Figure 5-1	PLCSIM Advanced Symbol.....	46
Figure 5-2	Example: Message in the taskbar .....	46
Figure 5-3	Control Panel .....	48
Figure 5-4	Example: Download via the "PLCSIM Virtual Ethernet Adapter" (TCP/IP) after naming.....	51
Figure 5-5	Structure of the MAC address for an instance .....	52
Figure 5-6	Add card reader .....	56
Figure 5-7	Preview of download dialog .....	57
Figure 7-1	External applications and Simulation Runtime.....	66
Figure 7-2	Access to instances with distributed communication .....	67
Figure 7-3	API and external applications.....	68
Figure 8-1	Policy exceptions for VMware vSphere Hypervisor (ESXi).....	309



# Guide

## 1.1 Documentation Guide S7 PLCSIM Advanced

The documentation for the SIMATIC S7-1500 automation system and the SIMATIC ET 200SP distributed I/O system is arranged into three areas.

### Basic information

System manuals and Getting Started describe in detail the configuration, installation, wiring and commissioning of the SIMATIC S7-1500 and ET 200SP systems. The STEP 7 online help supports you in configuration and programming.

### Device information

Product manuals contain a compact description of the module-specific information, such as properties, terminal diagrams, characteristics and technical specifications.

### General information

The function manuals contain detailed descriptions on general topics such as diagnostics, communication, Motion Control, Web server, OPC UA.

You can download the documentation free of charge from the Internet (<http://w3.siemens.com/mcims/industrial-automation-systems-simatic/en/manual-overview/Pages/Default.aspx>).

Changes and additions to the manuals are documented in product information sheets.

You will find the product information on the Internet:

- S7-1500 (<https://support.industry.siemens.com/cs/us/en/view/68052815>)
- ET 200SP (<https://support.industry.siemens.com/cs/us/en/view/73021864>)

## Manual Collections

The Manual Collections contain the complete documentation of the systems put together in one file.

You will find the Manual Collections on the Internet:

- S7-1500 (<https://support.industry.siemens.com/cs/ww/en/view/86140384>)
- ET 200SP (<https://support.industry.siemens.com/cs/ww/en/view/84133942>)

## "mySupport"

With "mySupport", your personal workspace, you make the best out of your Industry Online Support.

In "mySupport", you can save filters, favorites and tags, request CAx data and compile your personal library in the Documentation area. In addition, your data is already filled out in support requests and you can get an overview of your current requests at any time.

You must register once to use the full functionality of "mySupport".

You can find "mySupport" on the Internet (<http://support.industry.siemens.com/My/ww/en/documentation>).

## "mySupport" - Documentation

In the Documentation area in "mySupport" you can combine entire manuals or only parts of these to your own manual.

You can export the manual as PDF file or in a format that can be edited later.

You can find "mySupport" - Documentation on the Internet (<https://support.industry.siemens.com/My/ww/en/>).

## Application examples

The application examples support you with various tools and examples for solving your automation tasks. Solutions are shown in interplay with multiple components in the system - separated from the focus on individual products.

You will find the application examples on the Internet (<https://support.industry.siemens.com/sc/ww/en/sc/2054>).

## TIA Selection Tool

The TIA Selection Tool can be used to select, configure and order devices for Totally Integrated Automation (TIA). It assembles the configuration editors for automation technology already familiar into a single tool.

With the TIA Selection Tool, you can generate a complete order list from your product selection or product configuration.

You can find the TIA Selection Tool on the Internet (<http://w3.siemens.com/mcmts/topics/en/simatic/tia-selection-tool>).

## 1.2 S7 PLCSIM products

### S7 PLCSIM Advanced

The version of S7-PLCSIM Advanced supports the following SIMATIC CPU series:

- S7-1500, S7-1500F, S7-1500C, S7-1500T and S7-1500TF
- ET 200SP and ET 200SP F

#### Communication

S7-PLCSIM Advanced supports communication via Softbus or TCP/IP.

S7 PLCSIM Advanced enables C++ and C# programs and simulation software to access SIMATIC CPU series supported by S7-PLCSIM Advanced via the user interface (API).

S7-PLCSIM Advanced and S7-PLCSIM V12 cannot be installed on the same PC or the same virtual machine. The communication between the two applications cannot be simulated.

S7-PLCSIM V5.4 SP7 is automatically installed with S7-PLCSIM Advanced. The communication between the two applications can be simulated.

### S7-PLCSIM V14

The version of S7-PLCSIM supports the following SIMATIC CPU series:

- S7-1200 and S7-1200F
- S7-1500, S7-1500F, S7-1500C, S7-1500T and S7-1500TF
- ET 200SP and ET 200SP F

#### Communication

S7-PLCSIM V14 supports communication via Softbus.

### S7-PLCSIM V5.x

S7-PLCSIM V5.x simulates the following SIMATIC CPU series:

- S7-300 and S7-300F
- S7-400 and S7-400F

S7-PLCSIM V5.x cannot be installed on the same PC or the same virtual machine S7-PLCSIM as of V12.

S7-PLCSIM V5.4 SP7 is automatically installed with S7-PLCSIM Advanced.

#### Communication

S7 PLCSIM V5.x can communicate via Softbus with instances of S7-PLCSIM as of V12.

S7-PLCSIM V5.4 SP7 can communicate via Softbus with instances of S7-PLCSIM Advanced.



# Product overview

## 2.1 What is PLCSIM Advanced?

Simulation systems support the development of programs and the deployment in production that follows. In the automation world, a simulated test environment shortens commissioning times. It is possible to test the program after program changes in the virtual controller before it is loaded into the corresponding real controller and the plant is put into operation.

### S7 PLCSIM Advanced

Using S7-PLCSIM Advanced, you can simulate your CPU programs on a virtual controller. You do not need a real controller for this. You can configure your CPU in STEP 7 V14, program your application logic and then load the hardware configuration and the program into the virtual controller. From there you can run your program logic, observe the effects of simulated inputs and outputs and adapt your programs.

In addition to communicating via Softbus, S7-PLCSIM Advanced provides a full Ethernet connection and can thus also communicate distributed.

#### User interface (API)

S7 PLCSIM Advanced enables interaction with native C++/C# programs or software via the user interface.

### Application areas

- Software in the Loop-Simulation for virtual commissioning of machine tools
- In combination with third-party software:
  - Simulation of production machines and plants
  - Combined simulation of automation and mechanics

### Advantages

The use of S7 PLCSIM Advanced offers numerous advantages:

- Improve quality of automation projects
- Accelerate time to market
- Reduce production times
- Reduce risk for commissioning
- Avoid costs for hardware in simulation environments
- Increase efficiency in maintenance

## 2.2 Security for PLCSIM Advanced

### Restrictions for security

Note the following restrictions when using PLCSIM Advanced:

#### Authentication

- The user interfaces (API) do not have options for authentication and authorization. There is no protection using user accounts and passwords.
- The Runtime Manager communication is not protected by authentication.

#### Communication

- The multi-computer simulation communication is not encrypted.
- A TCP/IP port is opened on the PC for cross-network communication.
- The installed WinPCap component provides access to TCP/IP network communication.

---

#### Note

For cross-computer communication, it is recommended to use a closed simulation network that is not connected to a production network.

---

#### Know-how protection

---

#### Note

If know-how-protected blocks for the simulation support are enabled, the know-how protection is limited.

---

## 2.3 Simulations support

### Requirement for simulation

#### Note

#### Enable simulation capability

To use a STEP 7 project with simulation, you must select the "Support simulation during block compilation" option in the "Protection" tab in the project properties.



Figure 2-1 Enable simulation capability

### Know-how protection

If a know-how-protected block is to be used for the simulation, it must be unlocked by entering a password, to enable the "Simulation with SIMATIC S7-PLCSIM Advanced" option to be unlocked in the properties of the block in the tab "General > Compilation".

### Global libraries

You cannot use know-how protection with global libraries, because the libraries are write-protected.

The "Simulation with SIMATIC S7-PLCSIM Advanced" option must be set when generating the blocks (source of the blocks).

## 2.4 Supported CPUs

### Supported CPUs from the S7-1500 family

S7-PLCSIM Advanced V1.0 supports the simulation of the following CPUs (all firmware versions, V2.0 recommended):

Type		Article number
<b>Standard CPUs</b>	CPU 1511-1 PN	6ES7511-1AK01-0AB0
	CPU 1513-1 PN	6ES7513-1AL01-0AB0
	CPU 1515-2 PN	6ES7515-2AM01-0AB0
	CPU 1516-3 PN/DP	6ES7516-3AN01-0AB0
	CPU 1517-3 PN/DP <sup>3</sup>	6ES7517-3AP00-0AB0
	CPU 1518-4 PN/DP <sup>3</sup>	6ES7518-4AP00-0AB0
	CPU 1518-4 PN/DP ODK <sup>2, 3</sup>	6ES7518-4AP00-3AB0
<b>Fail-safe CPUs</b>	CPU 1511F-1 PN	6ES7511-1FK01-0AB0
	CPU 1513F-1 PN	6ES7513-1FL01-0AB0
	CPU 1515F-2 PN	6ES7515-2FM01-0AB0
	CPU 1516F-3 PN/DP	6ES7516-3FN01-0AB0
	CPU 1517F-3 PN/DP <sup>3</sup>	6ES7517-3FP00-0AB0
	CPU 1518F-4 PN/DP <sup>3</sup>	6ES7518-4FP00-0AB0
	CPU 1518F-4 PN/DP ODK <sup>2, 3</sup>	6ES7518-4FP00-3AB0
<b>Compact CPUs<sup>1</sup></b>	CPU 1511C-1 PN	6ES7511-1CK00-0AB0
	CPU 1512C-1 PN	6ES7512-1CK00-0AB0
<b>ET 200SP CPUs</b>	CPU 1510SP-1 PN	6ES7510-1DJ01-0AB0
	CPU 1510SP F-1 PN	6ES7512-1SJ01-0AB0
	CPU 1512SP-1 PN	6ES7512-1DK01-0AB0
	CPU 1512SP F-1 PN	6ES7512-1SK01-0AB0
<b>Technology CPUs</b>	CPU 1511T-1 PN	6ES7511-1TK01-0AB0
	CPU 1515T-2 PN	6ES7515-2TM01-0AB0
	CPU 1517T-3 PN/DP <sup>3, 4</sup>	6ES7517-3TP00-0AB0
	CPU 1517TF-3 PN/DP <sup>3, 4</sup>	6ES7517-3UP00-0AB0

<sup>1</sup> The on-board I/O within the compact CPUs is not simulated. The simulation interface corresponds to the process image.

<sup>2</sup> The ODK functionality of this CPU is not simulated.

<sup>3</sup> The simulation of this CPU only supports 5120 Motion Control resources.

<sup>4</sup> The simulation of this CPU supports only 64 cams.

### Unsupported CPUs

S7-PLCSIM Advanced V1.0 does not support the simulation of the following CPUs:

- S7-1200 CPUs
- ET 200pro, ET 200pro F CPUs
- ET 200SP Open Controller CPU 1515SP PC
- Software Controller

## 2.5 Differences between a simulated and a real CPU

The virtual controller cannot fully simulate a real CPU down to the individual details. Even if a program is downloaded without errors to the CPU and running successfully, this does not necessarily mean that the virtual controller in the simulation behaves exactly like a real CPU.

### Deterministic

S7-PLCSIM Advanced runs on a PC with the Windows operating system. Therefore, the scan cycle time and the exact time of actions in S7-PLCSIM Advanced are not the same as when these actions run on physical hardware. This is because that several programs share the processing resources on your PC.

If your program depends heavily on the time required to execute actions, then make sure that you do not evaluate your program based only on the results of the simulation time.

### Know-how protection

Projects with know-how protected blocks can only be simulated when they are enabled for simulation. You need the block password for this purpose.

### Instructions

Instructions are simulated with a few exceptions, for example, interrupts. Programs that are based on the instructions behave different than real CPUs in the simulation.

### See also

Restrictions for instructions (Page 306)

## 2.5.1 Restrictions for all supported CPUs

### I/O

S7-PLCSIM Advanced simulates the real CPU, but not configured I/O modules and the on-board I/O of the compact CPUs.

### Bus systems

S7-PLCSIM Advanced does not simulate bus systems (PROFINET IO, PROFIBUS DP, backplane bus).

### Process image partitions

S7-PLCSIM Advanced always updates the address areas of the process image at the cycle control point. Process image partitions between two cycle control points are not updated.

## Diagnostics

S7-PLCSIM Advanced cannot simulate the complete range of all the possible diagnostic buffer entries, for example, no entries that affect the real hardware. Most I/O and program errors are simulated.

## Online and diagnostic functions

Some online and diagnostic functions are not very useful in the simulation and are therefore not supported. These include, for example, the functions "Format memory card" and "Firmware update".

## Status indicators LED flashing

In STEP 7, you can have the LED displayed on an CPU flash using the "Extended download to device" dialog. S7-PLCSIM Advanced does not simulate this function.

## Data logging

S7-PLCSIM Advanced does not simulated data logging, because this function writes all output to the SD memory card and S7-PLCSIM Advanced does not support the use of an SD memory card.

## Recipes

S7-PLCSIM Advanced does not simulate the use of recipes.

## Backup and Restore

S7-PLCSIM Advanced does not simulate the product property "Backup and Restore".

## Limited support

S7-PLCSIM Advanced simulates some functions to a limited extent. You can find an overview in the section Restrictions (Page 302).

## 2.6 Simulating Motion Control

### Restrictions

PLCSIM Advanced supports STEP 7 projects with configurations and functions for motion control for the CPUs S7-1500, S7-1500F, S7-1500C, S7-1500T, S7-1500T F, ET 200SP and ET 200SP F.

PLCSIM Advanced provides only limited support for technology objects in some CPUs, see Restrictions with Motion Control (Page 307).

### Simulation with external simulation software

In a virtual S7-1500 controller, the technology objects are connected to the process image. Simulation software can thus access the process image via the user interfaces (API) of S7-PLCSIM Advanced and simulate the behavior of the other connected axes.

### Simulation mode in STEP 7

The simulation mode is a standard function of the technology objects.

If you want to move an axis in simulation mode, select the "Activate simulation" check box in STEP 7 under "Technology Object > Configuration > Basic Parameters > Simulation". No additional setting is required for a virtual axis.

### Feedback of the axis position

The speed setpoint of the simulated drive is integrated into the actual position value with a time delay (PT1). The result of this calculation is returned to the technology object as position actual value of the axis.

### Reference point approach of the axis

If you selected "Use zero mark via PROFIdrive frame" in STEP 7 for the reference point approach, PLCSIM Advanced responds immediately to any active (mode 2, 3, 8) or passive (mode 4, 5) reference point approach command (MC\_Home). The actual position is predefined as the reference point.

### Additional information

Information on "Setting in the drive and encoder connection" for actual value calculation of a virtual axis and on the topic "Virtual axis/Simulation" is available in the S7-1500T Motion Control (<https://support.industry.siemens.com/cs/ww/en/view/109481326>) function manual.

# Installing

## 3.1 Introduction

### 3.1.1 System requirements

You need high-performance computer hardware if you intend to run multiple instances of PLCSIM Advanced at the same time or to simulate communication between PLCSIM Advanced V1.0 and HMI devices as of version 14.0.

#### Minimum requirements for hardware and software

Preferably, you should install PLCSIM Advanced on a stand-alone PC, independent of STEP 7. Alternatively, you can install PLCSIM Advanced on the configuration PC on which STEP 7 is already installed.

For PLCSIM Advanced to operate efficiently, the computer hardware and software must meet minimum requirements.

Table 3- 1 Minimum requirements for hardware and software

Hardware / software	Requirement
Processor	2.2 GHz Intel® Celeron® Dual Core
RAM	<ul style="list-style-type: none"> <li>• 4 GB for one instance</li> <li>• 8 GB for 4 instances</li> </ul>
Free hard disk space	5 GB
Operating system 64-bit version	<ul style="list-style-type: none"> <li>• Windows 7 Home Premium SP1</li> <li>• Windows 7 Professional SP1</li> <li>• Windows 7 Enterprise SP1</li> <li>• Windows 7 Ultimate SP1</li> <li>• Windows Server 2012 R2 StdE (full installation)</li> </ul>
Screen resolution	1024 x 768

---

#### Note

Make sure that the Windows operating system you are using is up to date.

---



## Virtualization platforms

You can install STEP 7 and PLCSIM Advanced on a virtual machine. For this purpose, use one of the following virtualization platforms in the specified version or a newer version:

- VMware vSphere Hypervisor (ESXi) 6.0
- VMware Workstation 12 Pro
- VMware Workstation Player 12
- Microsoft Windows Server 2012 R2 Hyper-V

### Guest operating systems

You can use the following guest operating systems on the selected virtualization platform for the installation of STEP 7 V14 and PLCSIM Advanced:

- Windows 7 Professional SP1 / Ultimate SP1 / Enterprise SP1 (64-Bit)
- Windows Server 2012 R2 (64-Bit)

## 3.1.2 Licenses

### Floating license

S7 PLCSIM Advanced is supplied with a floating type license. These can be stored locally and shared for a network.

---

#### Note

A floating license is valid for **one** instance of a virtual controller within a PLCSIM Advanced installation.

---

### Handling licenses

You can learn how to handle the licenses for S7 PLCSIM Advanced instances on the DVD in the description of the SIMATIC Automation License Manager (ALM).

### API functions for licenses

- Return values for API function `PowerOn()` (Page 127) and callback function `OnOperatingStateChanged` (Page 228)
  - `SREC_OK` when a license is available.
  - `SREC_WARNING_TRIAL_MODE_ACTIVE` when no license is available, and an instance is started in a mode that allows unrestricted use of the instance for one hour.
- Return value for callback function `OnOperatingStateChanged`
  - `SREC_LICENSE_NOT_FOUND` when the instance is automatically shut down after expiration of the Trial mode.

## Warning in spite of existing license

---

### Note

A check is made for the presence of a license at power-up and after one hour during which you can test an instance without restriction (Trial Mode). When you import a license in the meantime, `SREC_WARNING_TRIAL_MODE_ACTIVE` is returned nevertheless.

---

### 3.1.3 Installation log

The log files contains automatically recorded information on the following installation processes:

- Installation of S7-PLCSIM Advanced
- Change or update of installation of S7-PLCSIM Advanced
- Repair of an existing installation of S7-PLCSIM Advanced
- Uninstallation of S7-PLCSIM Advanced

You can evaluate installation errors and warnings using the log files. You can troubleshoot the installation yourself or contact Siemens Technical Support. Product Support personnel need information from the installation log to analyze the problem. Send the folder with the log files as a ZIP file to Support.

### Memory location of the installation log

The memory location of the log file depends on the operating system. To open the folder with the log files, enter the environment variable "%autinstlog%" in the address bar in Windows Explorer. Alternatively, you reach the appropriate directory by entering "cd %autinstlog%" in the command line.

The log files are named as follows:

- SIA\_S7-PLCSIM\_Advanced\_V01@<DATE\_TIME>.log
- SIA\_S7-PLCSIM\_Advanced\_V01@<DATE\_TIME>\_summary.log

### Setup\_Report (CAB file)

An archive file with the installation log and all other required files is stored in CAB format. This archive file can be found at "%autinstlog%\Reports\Setup\_report.cab".

If you need help during installation, send this CAB file to SIEMENS Technical Support. Technical Support personnel can troubleshoot your installation based on the information in the CAB file.

A separate CAB file with a date ID is saved for each installation.

## 3.2 Installation medium

After installing S7-PLCSIM Advanced, keep the installation medium in a secure, easily accessible place.

You can use the installation medium to change, repair or uninstall, if necessary.

## 3.3 Installing PLCSIM Advanced

S7-PLCSIM Advanced starts the installation automatically when you insert the installation medium in your DVD drive.

### Installation requirements

Make sure that the following conditions are met before you begin the installation process:

- The hardware and software of the PC or Siemens Field PG meet the system requirements.
- The person who performs the installation has administrator rights on the respective computer.
- No other programs are active. This also applies to the Siemens Automation License Manager and other Siemens applications.
- All S7-PLCSIM versions V12 and higher are uninstalled.

---

#### Note

##### Security settings

For licensing via the ALM, when installing PLCSIM Advanced you must agree that port 4410 for TCP can be entered as an exception in the Windows Firewall (procedure step 7).

---

## Installing S7-PLCSIM Advanced

To install PLCSIM Advanced, follow these steps:

1. Insert the installation medium into the DVD drive of your computer. The setup program is automatically started, provided you have not deactivated the Autostart function on the Field PG or PC. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file. The "General settings" window is displayed.
2. The check box for the English is selected.
3. Click the "Read installation notes" button to display installation information. After you have read the notes, close the file.
4. Click the "Read Product Information" button to display the "Readme file". After you have read the information, close the file.
5. Click the "Browse" button if you want to change the default installation path. The installation path must not exceed 89 characters. The path name must not contain any UNICODE characters. If you select a different installation path than the default installation path, the desktop icon may not be displayed correctly.
6. Click "Next". The window with the security settings is displayed. To continue the installation, select the check box at the bottom of the screen to accept changes to the security and permissions settings of your system.
7. Click "Next". The window with the installation settings is displayed. You can save or print a report of the settings by clicking "Save report" or "Print report". Check the settings for correctness. If you want to make any changes, click "Back" until you reach the point in the installation process where you want to make changes. Once you have completed your changes, click "Next".
8. Your installation details are displayed in the overview screen. Click the "Install" button. The installation is then started and PLCSIM Advanced is installed on your computer.
9. After completion of the setup program, you must restart your computer. Select "Yes, I want to restart the computer now" to restart the computer immediately or select "No, I will restart computer later" to restart the computer later.
10. Click "Restart". If the computer is not restarted, click "Finish".

## Error during installation of S7-PLCSIM Advanced

When S7-PLCSIM Advanced is installed, any existing installation of S7-PLCSIM is displayed.

A requirement for installation of S7-PLCSIM Advanced is that no other S7-PLCSIM installation is located on the same computer.

Even though no installation of S7-PLCSIM is displayed in the "Programs and Features" list, it is still possible that the computer has an existing installation.

### Remedy

Run the setup for S7-PLCSIM V12 or V12 SP1 and uninstall the program.

When the setup is not available, download the setup files for S7-PLCSIM via Siemens Mall (<https://support.industry.siemens.com/cs/document/65601780>).

## 3.4 Changing PLCSIM Advanced

### Requirements for changing the installation

The following conditions must be met before you can start changing the installation:

- The hardware and software of the computer meet the system requirements.
- You have administrator rights on the installation computer.
- No other programs are active.

### Procedure for changing the installation

To change your PLCSIM Advanced installation, follow these steps:

1. Insert the installation medium into the drive. The setup program starts up automatically, provided you have not deactivated the Autostart function on the Field PG or PC.  
If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
2. Follow the prompts until you reach the "Configuration" window.
3. Select the "Change upgrade" check box.
4. Follow the remaining prompts to change your installation.
5. Complete the installation operation by restarting your computer.

---

**Note****Target directory**

You cannot change the target directory because you are changing an existing installation.

---

## 3.5 Repairing PLCSIM Advanced

If your installation becomes damaged, you can repair it with the PLCSIM Advanced data storage medium.

### Requirements for repairing the installation

The following conditions must be met before you can start repairing the installation:

- The hardware and software meet the system requirements.
- You have administrator rights on the installation computer.
- No other programs are active.

### Procedure for repairing the installation

To repair your installation, follow these steps:

1. Insert the installation medium into the drive. The setup program is automatically started, provided you have not deactivated the Autostart function on the Field PG or PC. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
2. Follow the prompts until you reach the "Configuration" window. Select the "Repair" check box.
3. Follow the remaining prompts to repair your installation.
4. Complete the repair operation by restarting your computer.

## 3.6 Uninstalling PLCSIM Advanced

You have two options for uninstalling S7-PLCSIM Advanced on your computer:

- You uninstall the program using the Windows Control Panel.
- You uninstall the full product using the installation medium.

### Uninstalling PLCSIM Advanced using the Windows Control Panel

To uninstall PLCSIM Advanced using the Windows Control Panel, follow these steps:

1. Double-click the "Programs and Features" option in the Windows Control Panel.
2. Right-click on "Siemens S7-PLCSIM Advanced V1.0" and select "Uninstall".
3. Follow the prompts for uninstallation.
4. Complete the uninstallation operation by restarting your computer.

If you do not perform a restart, the Runtime Manager continues running.

If problems occur when uninstalling PLCSIM Advanced using the Windows Control Panel, use the installation medium for uninstalling.

### Uninstalling PLCSIM Advanced using the installation medium

To uninstall PLCSIM Advanced using the installation medium, follow these steps:

1. Insert the installation medium into the drive. The setup program is started automatically, provided you have not deactivated the Autostart function on the programming device or PC. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.

If you do not perform a restart, the Runtime Manager continues running.

2. Follow the prompts until you reach the "Configuration" window. Your previous installation is detected. Select the "Uninstall" check box.
3. Follow the prompts to uninstall PLCSIM Advanced.
4. Complete the uninstallation operation by restarting your computer.

If you do not perform a restart, the Runtime Manager continues running.

### Uninstalling additional software

When PLCSIM Advanced is uninstalled, the following software remains installed:

- Automation License Manager
- S7-PLCSIM V5.4
- .NET Framework
- WinPcap

If you also want to uninstall this software, use the Windows Control Panel.

## Communication paths

### Local and distributed communication

The following paths are open for communication between STEP 7 V14 and the instances of PLCSIM Advanced user interfaces:

Communication paths	Local	Local	Distributed
Protocol	Softbus	TCP/IP	TCP/IP
Communication interface in PLCSIM Advanced	PLCSIM	PLCSIM Virtual Ethernet Adapter	PLCSIM Virtual Ethernet Adapter
STEP 7 and instances	on a PC / VM	on a PC / VM	distributed
<b>Communication...</b>			
between STEP 7 and instances	Yes	Yes	Yes
among instances	Yes	Yes	Yes
possible via OPC UA server and Web server	No	Yes	Yes
between an instance and a real hardware CPU	No	No	Yes
between an instance and a real HMI V14	No	No	Yes
between an instance and a simulated HMI V14	Yes	Yes	No

### Softbus

Softbus is a communication path via a virtual software interface.

The communication is limited to a local PC or a virtual machine. The advantage here is that no data can be accidentally downloaded to a hardware CPU or communicate with real hardware.

### Select communication interface

You program the communication interface via the user interface (API) or select it in the Control Panel under "Online Access". The setting is valid for all generated instances. The default setting is the communication via "PLCSIM" (Softbus).

Additional network settings are necessary for the distributed communication via the "PLCSIM Virtual Ethernet Adapter" (TCP/IP).



### API functions for selecting the communication interface

- GetCommunicationInterface() (Page 116)
- SetCommunicationInterface() (Page 116)
- CommunicationInterface { get; set; } (Page 117)

### See also

Interfaces - Information and settings (Page 113)  
S7 PLCSIM Advanced Control Panel (Page 46)

## 4.1 Local communication

Local communication can be performed via the Softbus protocol or TCP/IP.

For local communication, the PLCSIM Advanced instance is on the same PC or on the same virtualization platform (VMware) as STEP 7 or another communication partner.

### Local communication via Softbus

For security reasons, the local communication is performed via Softbus in PLCSIM Advanced by default.

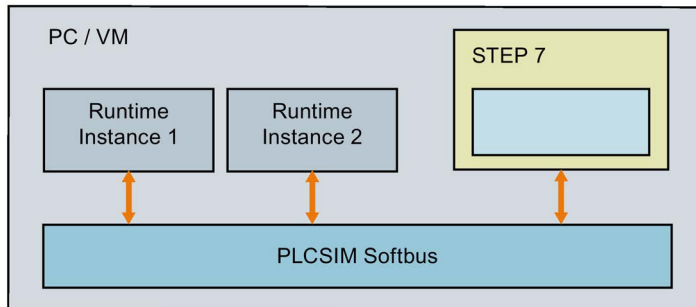


Figure 4-1 Local communication via Softbus

### Local communication via TCP/IP

Communication is performed via the PLCSIM Virtual Ethernet Adapter, a virtual network interface that behaves like a real network interface.

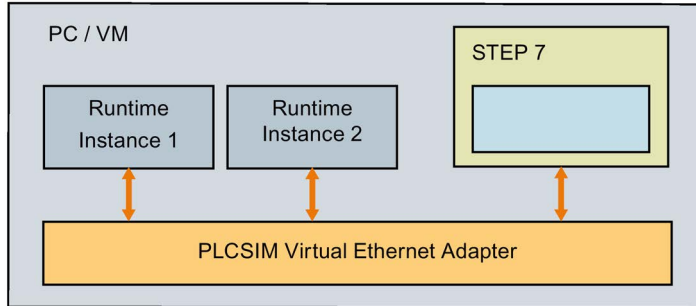


Figure 4-2 Local communication via TCP/IP

## 4.2 Communication via TCP / IP

### Distributed communication

Distributed communication via TCP/IP means that the PLCSIM Advanced instances communicate with the other devices via the Virtual Switch. Communication is possible with real or simulated CPUs, real or simulated HMIs.

The PLCSIM Virtual Switch must be activated on the PLCSIM Virtual Ethernet Adapter for instances on the network to be visible.

### Example 1: Distributed communication

In the following example, STEP 7 is on a PC and the PLCSIM Advanced instances are on another PC. The PCs are connected via the Ethernet adapter.

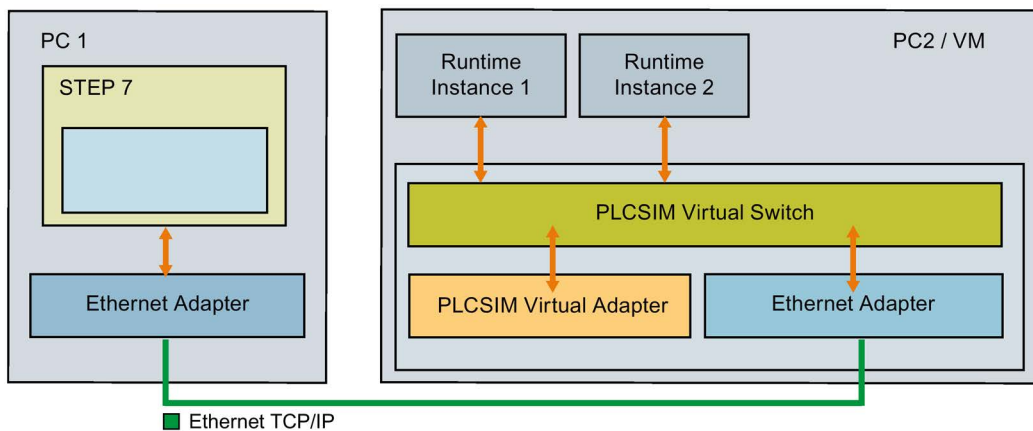


Figure 4-3 Distributed communication via Ethernet

### Example 2: Distributed communication

In the following example, STEP 7 is on a PC and the PLCSIM Advanced instances are on a virtual machine on the same PC. PC and virtual machine are connected via the network adapters.

#### Recommendation

Use to ensure the VMware settings as the network adapter type for the Bridged Mode to ensure error-free operation.

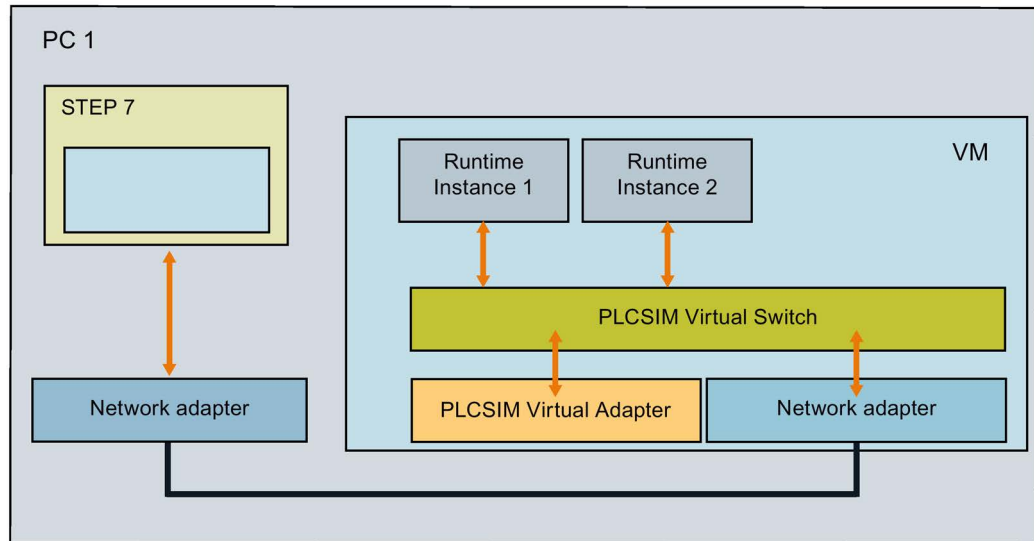


Figure 4-4 Distributed communication via network adapters

## 4.3 Enable distributed communication

By default, the PLCSIM Virtual Switch can only communicate locally. For a distributed, i.e. multi-computer, communication to be possible, you must activate the PLCSIM Virtual Switch for a real network adapter.

#### Note

##### Network adapter

Make sure that only one network adapter of the PLCSIM Virtual Switch is activated.

The Control Panel of PLCSIM Advanced checks the activation and may report an incorrect configuration (error code -50).

### Activate PLCSIM Virtual Switch

To make the PLCSIM instances visible on the network and to reach other devices, activate the PLCSIM Virtual Switch in the Control Panel of PLCSIM Advanced or under Windows:

1. To do this, open the "Network and Sharing Center" in the Windows Control Panel.
2. Open the properties of the desired network adapter, for example, for the "Local Area Connection".
3. Select the check box for the "Siemens PLCSIM Virtual Switch" and confirm with OK.

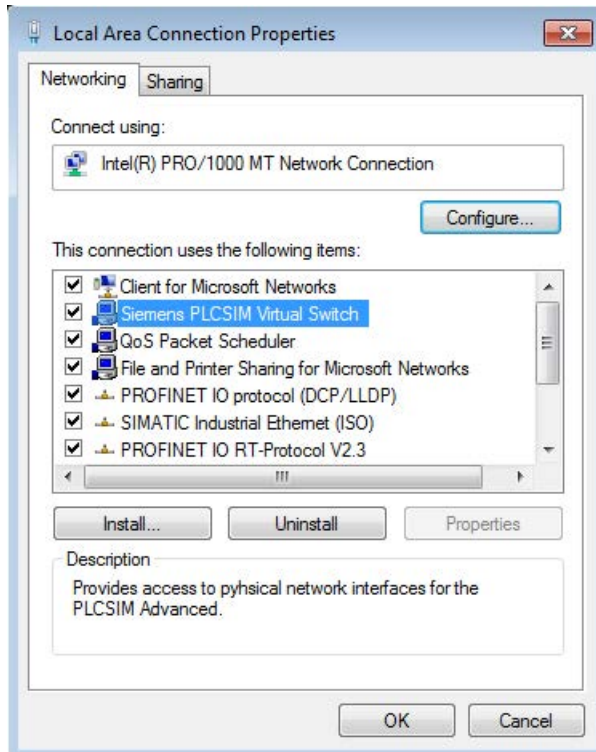


Figure 4-5 Activate PLCSIM Virtual Switch

### Accessible devices

When the PLCSIM Virtual Switch is activated, STEP 7 shows the devices available on the Virtual Ethernet Adapter in the project tree.

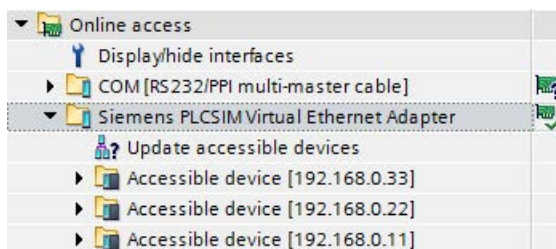


Figure 4-6 Accessible devices on the Virtual Ethernet Adapter

# Simulate CPU

## 5.1 Basic procedure for the simulation

The following overview shows the basic steps to perform simulation with an instance of a virtual controller.

### Requirements

The following requirements must be met for starting simulation via local communication:

- STEP 7 V14 and S7 PLCSIM V1.0 Advanced are installed on the same PC.
- The CPU hardware is configured in STEP 7.

---

#### Note

##### Enable simulation support

In the "Protection" tab in the properties of the project in STEP 7, select the check box "Support simulation during block compilation".

---

### Create and activate an instance via the Control Panel

- Open PLCSIM Advanced Control Panel
- Open the "Start Virtual S7-1500 PLC" options
- Enter a name for an instance
- Select CPU type
- Create an instance using the "Start" button

### In STEP 7, perform the download and start the simulation

- Download the program to the virtual controller
- Switch the controller to RUN to start the simulation
- Perform diagnostics
- ...

### See also

Simulations support (Page 27)

## 5.2 User interface

### 5.2.1 S7 PLCSIM Advanced Symbol

After installing PLCSIM Advanced, the following icons are on the Windows desktop:



Figure 5-1 PLCSIM Advanced Symbol

After double-clicking the icon, the icon appears in the taskbar of the information area.

You can use Windows functions to permanently display the icon in the information area of the taskbar.

#### Opening a graphical interface

Right-clicking on the icon in the taskbar opens the graphical interface of PLCSIM Advanced, the Control Panel.

If the Control Panel is open, you can use the mouse-over function to display messages about the current status of the instances.

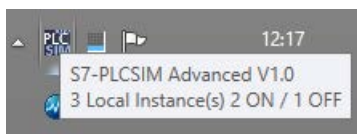
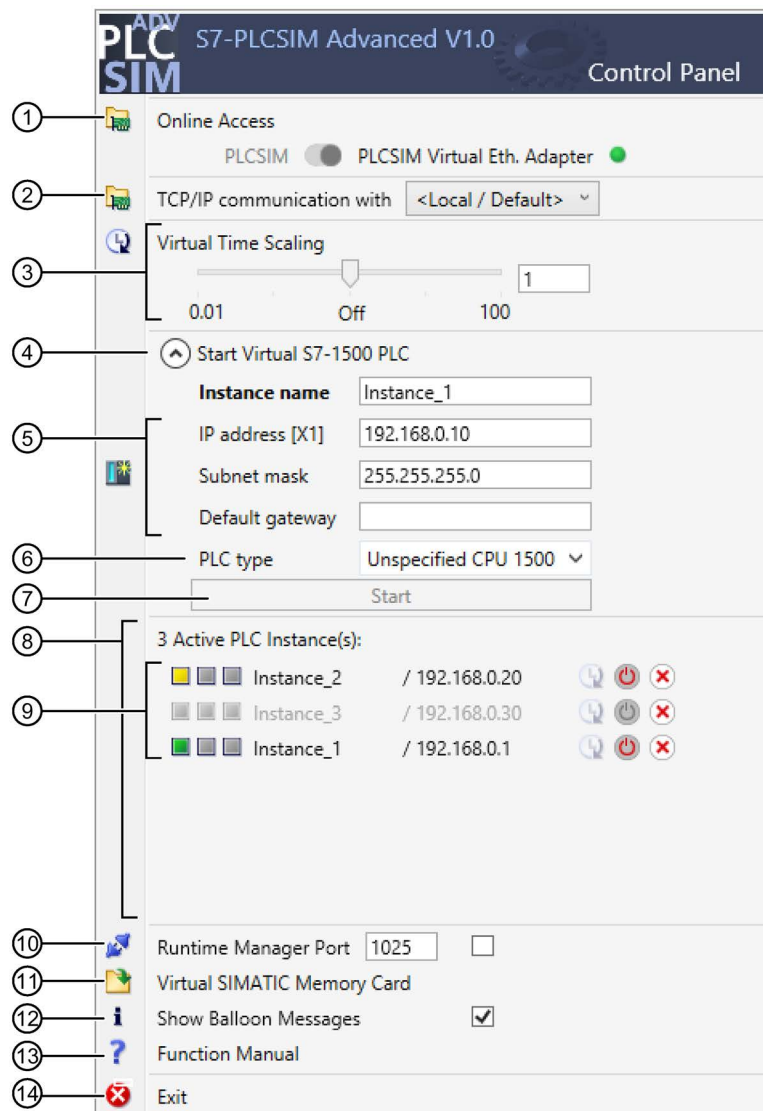


Figure 5-2 Example: Message in the taskbar

### 5.2.2 S7 PLCSIM Advanced Control Panel

#### Control Panel

The Control Panel is optional and is not needed to use PLCSIM Advanced via the API. It is available in English in version 1.0.



- |   |   |   |
|---|---|---|
| ① | Online access   | Switch to select the communication interface  |
| ② | TCP/IP communication  | Selection of network adapter for distributed communication  |
| ③ | Virtual time  | Slider to adjust the scaling factor   |
| ④ | <b>Start Virtual S7-1500 PLC</b>  |   |
|   | Name of the instance  | Here you enter a unique name for the instance. Enter a minimum of 3, a maximum of 64 characters. If the name is unique in the network, the "Start" button is enabled. |
| ⑤ | <ul style="list-style-type: none"> <li>• IP address</li> <li>• Subnet mask</li> <li>• Standard gateway</li> </ul> | The input boxes are visible when you switch the communication interface to "PLCSIM Virtual Ethernet Adapter".   |
| ⑥ | CPU type  | Here you select the type of CPU to be simulated.  |
| ⑦ | "Start" button  | Create with the button and start an instance with the entered name.   |
| ⑧ | Instance list   | The list shows the available <b>local</b> instances.  |
| ⑨ | LED displays  | The meaning of the LED is displayed when you move the mouse over it.  |

⑩	Runtime Manager Port	Here you open a port on the local PC.
⑪	Virtual SIMATIC Memory Card	Open an Explorer window here with the path to the virtual memory card.
⑫	Display messages	Here you disable the PLCSIM Advanced messages in the Windows task bar for the duration of the operation.
⑬	Function manual	This is where you open the S7 PLCSIM Advanced Function Manual in a standard PDF viewer.
⑭	Exit	Exit switches all instances off and closes the Control Panel.

Figure 5-3 Control Panel

### Switch for communication interface

Use the switch to select the communication interface for all instances to be created:

- "PLCSIM corresponds to the local communication via Softbus (default).
- "PLCSIM Virtual Ethernet Adapter corresponds to the communication via TCP/IP.

The setting applies to all other instances. The selected communication interface for starting an instance is maintained until all instances are shut down.

When an instance is already started, it sets "its" communication interface as the default for other instances.

#### Change communication interface

To change the communication interface, switch off all instances and enable the other interface.

### TCP/IP communication

You can select a real network adapter from the drop-down list during operation. You thus activate the PLCSIM Virtual Switch and establish TCP/IP communication between the instances and the real network.

The <Local> setting disables the PLCSIM Virtual Switch and disconnects the instances from the real network. Only local TCP/IP communication over virtual adapter is possible in this case.

### Virtual time

Use the slider or the mouse wheel to select the scaling factor for the virtual time.

The selected scaling factor applies to the instances for which the virtual time is enabled.

Clicking on "Off" restores the default (1) again. For more information, see Virtual time response (Page 58).



## Create instance (local)

To create an instance, enter a unique name under "Instance Name". If the name already exists in the directory of the Virtual SIMATIC Memory Card, the existing instance is started.

In the "PLC-Type" drop-down list, select the Unspecified CPU 1500 or Unspecified ET 200SP CPU type. The CPU is named with the first download of the TIA Portal.

**Recommendation:** Create up to 4 local instances on your PC or visualization platform. Remote instances cannot be created through the Control Panel.

## Instance list

The instance list contains the instances that are started locally on the PC or virtualization platform. Instances that have already been started on the runtime API are detected and displayed in the list.

The LED displays show the status of the instance that corresponds to those of the hardware CPU.

RUN and STOP are displayed depending on the current operating state of the instance.

You can "operate" the instance with icons:

 Apply scaling factor for the virtual time,  disable virtual time,

 Switch on instance,  Switch off instance,

 Switch off instance and log out of the Runtime Manager

## Runtime Manager Port

A remote connection can be established to another Runtime Manager via the specified port. The value must be greater than 1024.

If you select the check box, the port remains stored. You can use the remote connection without having to make this setting every time you start the Control Panel. To use this functionality, the Control Panel must be started and running in the background.

## Virtual SIMATIC Memory Card

The user program, the hardware configuration and the retentive data are stored on the Virtual SIMATIC Memory Card. Click the link to open the directory.

Default path: ...\\Documents\\Siemens\\Simatic\\Simulation\\Runtime\\Persistence

## Display messages

Each time the panel starts, help information and messages relating to the Control Panel are displayed, for example, when changing the IP address or when a license is missing. Disable the display if you do not need the messages.

## Exit

- The command switches off all local instances on the PC or the VM and logs them off from the Runtime Manager.
- This command closes the Runtime Manager if there is no remote connections to other Runtime Managers.
- If the Runtime Manager has remote connections to instances on additional PCs, these instances and the Runtime Manager continue to run.

## Minimize Control Panel

Clicking on an empty area on the desktop minimizes the Control Panel. The instances are not affected.

# 5.3 Download

## Requirements

You can download the STEP 7 project to the virtual controller when the following conditions are met:

- The instance is created via the Control Panel.
- The check box "Support simulation during block compilation" is selected.

## Select communication interface

In the Download dialog box, select the PG/PC interface:

- "PLCSIM" for download via Softbus
- "Siemens PLCSIM Virtual Ethernet Adapter" for download via TCP/IP

## Display in the download dialog

The dialog in STEP 7 at the first download of the CPU shows the compatible PLCSIM Advanced instances.

If the instance has not yet been configured after the first download only **one** interface is visible and it appears with the device type "CPU-1500 Simulation".

If the instance has been configured, the number of interfaces visible is determined by the number the CPU type has.

The list shows the interfaces of an instance with their IP addresses.

## Perform download

1. Select the PG/PC interface.
2. Click "Download".
  - In the "Load preview" window, STEP 7 shows the message "The downloads are performed on a simulated CPU".
  - After the first download, the PLCSIM Advanced instance displays the CPU type.

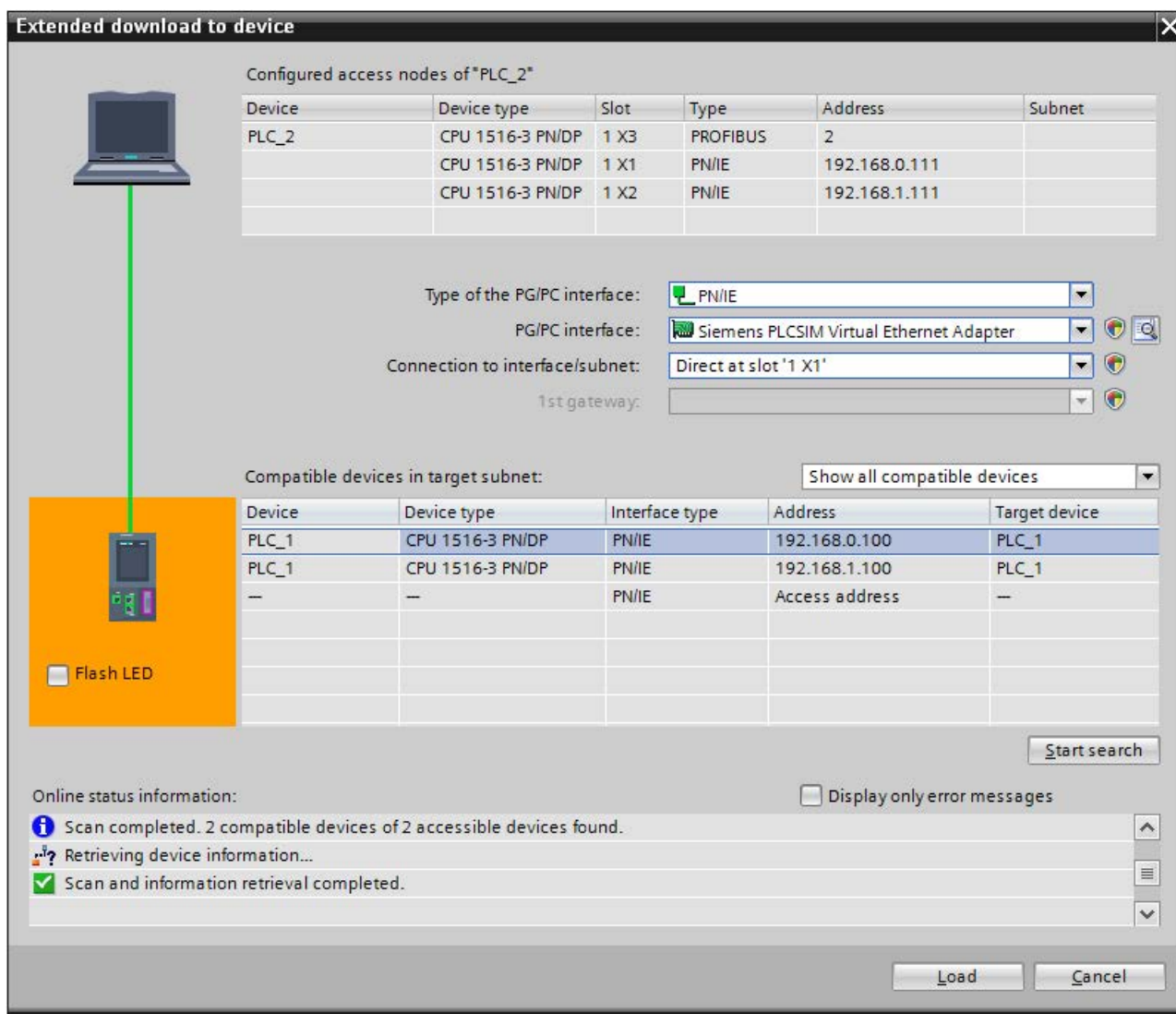


Figure 5-4 Example: Download via the "PLCSIM Virtual Ethernet Adapter" (TCP/IP) after naming

## 5.4 MAC address of the instances

### Detect CPUs and instances

If Ethernet interfaces of CPUs and PLCSIM Advanced instances are mixed in a network, the instances can be recognized by the "PLCSIM" suffix on the station type.

### Structure of the MAC address for an instance

The following figure shows the structure of the dynamically generated, locally managed MAC address:

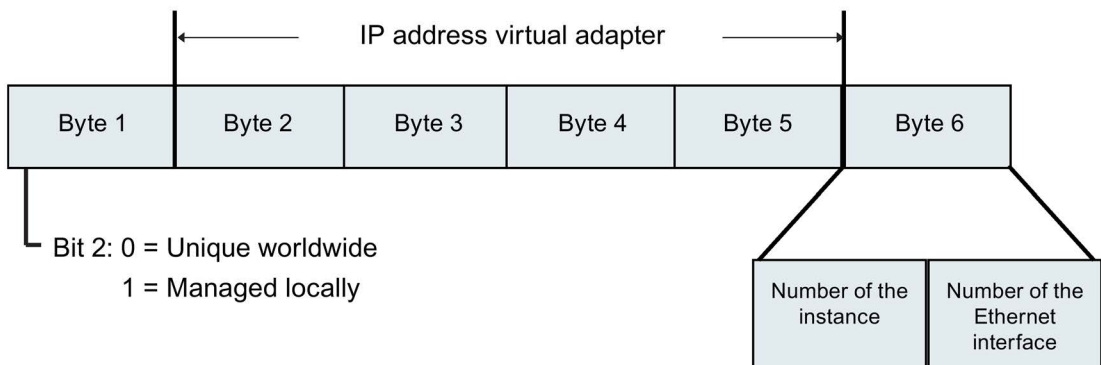


Figure 5-5 Structure of the MAC address for an instance

The MAC address tells you the PC on which a PLCSIM Advanced instance has been started.

### Assignment of the Ethernet interfaces

Port configurations of the Ethernet interfaces cannot be simulated in PLCSIM Advanced V1.0. Topological interconnection is not supported. A MAC address for a port is reserved internally for each Ethernet interface.

Table 5- 1 Assignment of the Ethernet interfaces, for example, a CPU 1518-4 PN/DP

Ethernet interface	Last digit of the MAC Address
IE 1	.....0
IE 1 / Port 1	.....1
IE 2	.....2
IE 2 / Port 1	.....3
IE 3	.....4
IE 3 / Port 1	.....5

## Example

02-C0-A8-00-83-10 means:

02 → locally managed MAC address of a PLCSIM Advanced instance

C0-A8-00-83 → IP of the Siemens PLCSIM Virtual Ethernet adapter = 192.168.0.131

1 → Instance 1

0 → Ethernet interface IE 1

If no Virtual SIMATIC Memory Card is loaded during startup of PLCSIM Advanced, the interfaces of PLCSIM Advanced display instances with their locally managed MAC address.

## 5.5 Simulate peripheral I/O

The Runtime API writes to and reads from a memory area. This memory is synchronized with the internal process image of the virtual S7-1500 controller at the cycle control point. The direct I/O accesses are made to this memory area. Only one process can access this memory at a given time.

The virtual controller must be in RUN to apply changes made by the API.

---

### Note

#### Dominance of the API when synchronizing

The API dominates when synchronizing. If the user program writes to the same address range as the API, the changes of the API overwrite those of the virtual controller.

---

## See also

Deviating I/O values in the STEP 7 user program (Page 310)

## 5.6 Simulate communication

### 5.6.1 Communication services that can be simulated

S7-PLCSIM Advanced V1.0 supports the following communication options:

Table 5- 2 Supported communication options

Communications options	Functionality / instructions
PG communication	On commissioning, testing, diagnostics
Open communication using TCP/IP	<ul style="list-style-type: none"> <li>• TSEND_C / TRCV_C</li> <li>• TSEND / TRCV</li> <li>• TCON<sup>1</sup></li> <li>• T_DISCON</li> </ul>
Open communication using ISO-on-TCP	<ul style="list-style-type: none"> <li>• TSEND_C / TRCV_C</li> <li>• TSEND / TRCV</li> <li>• TCON</li> <li>• T_DISCON</li> </ul>
Open communication via UDP <sup>2</sup>	<ul style="list-style-type: none"> <li>• TUSEND / TURCV</li> <li>• TCON</li> <li>• T_DISCON</li> </ul>
Communication via Modbus TCP	<ul style="list-style-type: none"> <li>• MB_CLIENT</li> <li>• MB_SERVER</li> </ul>
E-mail <sup>2</sup>	<ul style="list-style-type: none"> <li>• TMAIL_C</li> </ul>
S7 communication	<ul style="list-style-type: none"> <li>• PUT / GET</li> <li>• BSEND / BRCV</li> <li>• USEND / URCV</li> </ul>
OPC UA Server <sup>2</sup>	Data exchange with OPC UA clients
Web server <sup>2</sup>	Data exchange via HTTP

<sup>1</sup> When the "PLCSIM" interface (Softbus) is set, communication is performed **internally** via ISO-on-TCP.

<sup>2</sup> Only via the communication interface "PLCSIM Virtual Ethernet Adapter" (TCP/IP)

Special conditions apply when communicating with TUSEND/TURCV, see Restrictions for communications services (Page 305).

## 5.6.2 Communication between instances

PLCSIM Advanced supports communication between instances. An instance may be a simulation in PLCSIM Advanced V1.0 or a simulation in WinCC Runtime as of V14.

You can run two instances of PLCSIM Advanced, which then communicate with each other. To enable instances to communicate with each other, they must have a unique IP address.

### Each simulated CPU requires a unique IP address

If the CPUs have the same IP address, you cannot run multiple simulations. Each simulated CPU requires a unique IP address.

Make sure that the IP addresses in STEP 7 are unique before you start your simulations.

### T-block instructions and UDP

S7-PLCSIM Advanced simulates T-block connections for which the UDP protocol is configured only via the communication interface "PLCSIM Virtual Ethernet Adapter" (TCP/IP).

### T-block instructions and data segmentation

S7-PLCSIM Advanced implements T-block instructions with a data segmentation of 4 KB. A real CPU has data segmentation of 8192 bytes.

If you send more than 4 KB in a single TSEND instruction and receive data in ad hoc mode with a TRCV instruction, the TRCV instruction generates new data with only 4 KB. You must perform the TRCV instruction several times to receive additional bytes.

## 5.7 Provide project data offline for simulation

### Simulations regardless of STEP 7

To perform simulations independent of STEP 7, you can save the user program and the hardware configuration in STEP 7 in a directory.

### Provide project data offline

1. Create a "User-defined Card Reader" for your project data under Card Reader/USB storage in the project tree of STEP 7 for the CPU.
2. In the "Load preview" dialog for the target device, select "PLC Simulation Advanced" as an action, click in the selection field for this.  
→ The project is saved to the <Virtual Memory Card>\SIMATIC.S7S\OMSSTORE directory.
3. Save the folder \SIMATIC.S7S with the project data to a medium of your choice.

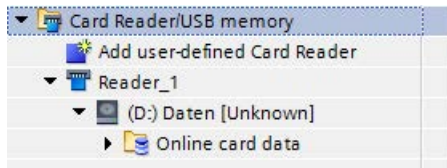


Figure 5-6 Add card reader



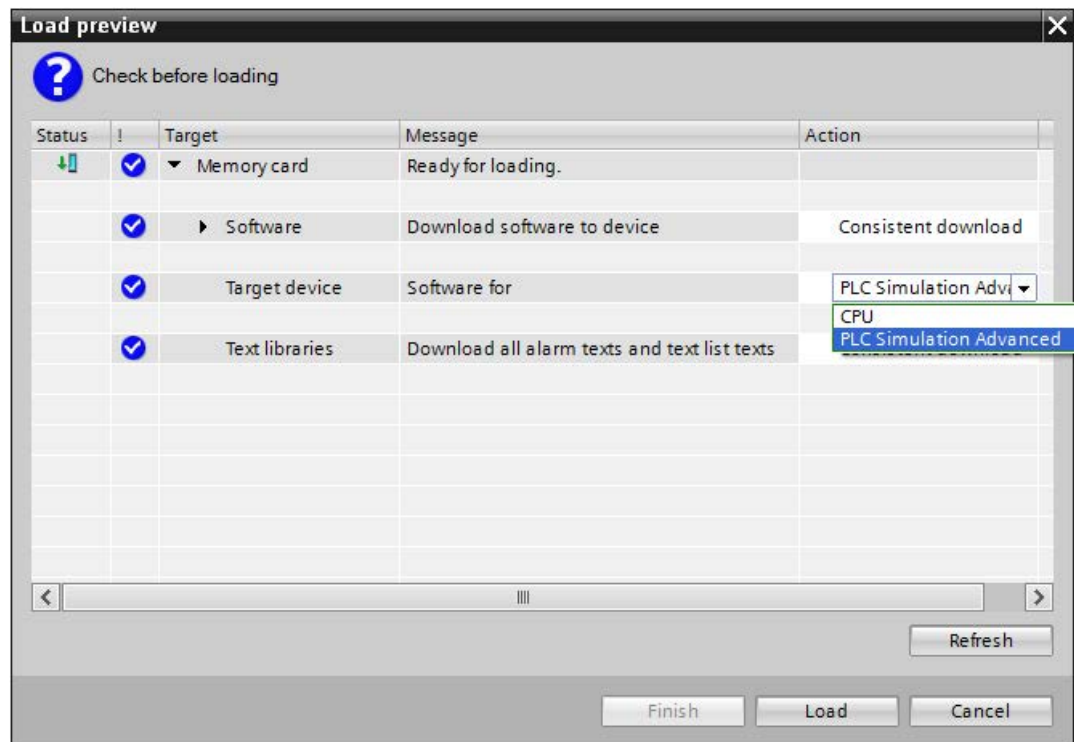


Figure 5-7 Preview of download dialog

## Provide project data for simulation

1. On the PC on which PLCSIM Advanced is installed, create the directory \SIMATIC\_MC in the directory in which the instance saves its data.
2. Move the \SIMATIC.S7S folder to the directory you have created.  
→ The instances can be started with the project data.

## API functions

The project data can be used for an instance via the user interface. Use of the following functions for this:

### API functions

- GetStoragePath() (Page 123)
- StoragePath { get; set; } (Page 124)

## See also

Controller - Information and settings (Page 119)

## Virtual time response

The virtual controller uses internally two types of clocks for simulation: A virtual clock and a real clock. The virtual clock is always the basis for the user program. It is used by components that are relevant for running the STEP 7 user program, such as cyclic OBs, cycle time monitoring, minimum cycle time, virtual system time and time calculations. Also, the time between two cycle control points is measured in virtual time.

The virtual time can be accelerated or slowed for test purposes.

The real clock always runs unchanged. It is used by components that are not subject to control processes, for example, communication with STEP 7.

### Virtual system time

When you start PLCSIM Advanced, the virtual system time of the virtual controller starts with the system time of Windows.

The virtual system time is based on the virtual clock, i.e. if a scaling factor is used, the system time runs correspondingly faster or slower.

All events that the virtual controller sends to the API provides a time stamp based on the system time.

---

#### Note

##### Difference between system time and local time

- System time: UTC  $\pm$  0 with daylight saving / standard time
  - Local time: UTC  $\pm$  time zone with daylight saving time / winter time
- 

#### API functions

- GetSystemTime() (Page 218)
- SetSystemTime() (Page 218)
- SystemTime { get; set; } (Page 218)

### Time offset

---

#### Note

Keep in mind that the time information of virtual system time and real local time differs by the time offset that is formed in addition to the selected scaling factor from the time zone offset and the daylight saving time/standard time offset.

---

## Scaling factor

Using a scaling factor, you can speed up or slow down the virtual clock of the virtual controller for simulations.

- The default is 1, i.e. the course of the virtual time corresponds to the course of real time.
- **Fast forward:** A scaling factor greater than 1 accelerates the virtual clock.  
Example: Scaling factor 2.0 → The virtual time is running twice as fast.
- **Slow motion:** A scaling factor less than 1 decelerates the virtual clock.  
Example: Scaling factor 0.5 → The progress of the virtual time slows down to 50%.

### API functions

- `GetScaleFactor()` (Page 219)
- `SetScaleFactor()` (Page 219)
- `ScaleFactor { get; set; }` (Page 220)

## See also

Settings for the virtual time (Page 218)

## 6.1 Speed up and slow down simulation

### Influence of fast forward and slow motion

Simulations can be accelerated and slowed down. Fast forward and slow motion only affects time-based components, for example, cyclic OBs. Compared to the real time, they are performed more frequently with fast forward and less frequently with slow motion.

Fast forward and slow motion do not change the execution speed of the CPU machine codes. For example, the speed at which all operations of an OB1 cycle are executed does not change. The execution speed depends on the processor of the PC on which the virtual controller is running. If you change the scaling factor, more or fewer cycle control points are reached in a given period of virtual time.

### Fast forward

To speed up the virtual time, select a scaling factor greater than 1 in the Control Panel or in the API.

---

#### Note

##### CPU load of the PC

If the scaling factor is greater than 1, the CPU load of the PC on which the virtual controller is simulated increases considerably.

---

#### Note

##### Performance

The performance is depends on the size of your project, among other things.

If the scaling factor is too high and the cycle-time monitoring indicates that the PC was incapable of calculating the OB1 or cyclic OBs in the specified time, the virtual controller goes to STOP.

**Recommendation:** To avoid this, start with a small scaling factor and gradually increase it step-by-step while keeping the virtual controller in RUN.

---

### Slow motion

To slow down the virtual time, select a scaling factor less than 1 in the Control Panel or in the API.

### API functions

- GetScaleFactor() (Page 219)
- SetScaleFactor() (Page 219)
- ScaleFactor { get; set; } (Page 220)

**See also**

Cycle control (Page 221)

Settings for the virtual time (Page 218)

Error with overflow cyclical events (Page 309)

## 6.2 Stop simulation

### Freeze state of the virtual controller

To stop a simulation and to synchronize a simulation partner, a virtual controller can be set to a freeze state via the user interface.

A triggered freeze state stops the virtual controller on the next cycle control point. The following occurs:

- The virtual time is stopped.
- No OBs running, not counters.
- The user program is no longer executed.
- The virtual controller is still accessible from the TIA Portal.
- The virtual controller is in a state consistent with input and output data.

---

**Note****Freeze-state ≠ operating state**

The freeze state is an internal operating state of the virtual controller. It does not correspond to RUN/STOP mode of a CPU. In the freeze state, the virtual controller maintains the last operating state.

- The LED display on the Control Panel and on the Web server accordingly shows RUN or STOP for instance.
  - The instance shows `SROS_FREEZE / Freeze`, see `EOperatingState` (Page 288).
-

## Trigger freeze state

To trigger the freeze state, following modes are available for the virtual controller:

- **Cycle-controlled operating mode** SingleStep and ExtendedSingleStep  
See Synchronize simulation partner cycle-controlled (Page 62).
- **Time-controlled mode** TimespanSynchronized  
See Synchronize simulation partner time-controlled (Page 64).

In **Default** mode, the virtual controller does not go into a freeze state.

### API functions

- Settings for the cycle control (Page 221)
- GetOperatingMode() (Page 221)
- SetOperatingMode() (Page 221)
- OperatingMode { get; set; } (Page 221)

## 6.3 Synchronize simulation partner

### 6.3.1 Synchronize simulation partner cycle-controlled

#### Introduction

The virtual controller must be set to the freeze state for synchronization.

The SingleStep and ExtendedSingleStep modes of the virtual controller provides one way to synchronize several simulation partners (clients).

#### SingleStep operating mode

In this operating mode, the virtual controller at the cycle control point goes into a freeze state and sends an event to the API clients.

The freeze state is terminated when the virtual controller issues an API command, performs the next step or changes to the default mode.

#### API functions

- RunNextCycle() (Page 225)

## ExtendedSingleStep operating mode

In this operating mode, the virtual controller at the cycle control point goes into a freeze state and sends an event to the API clients.

Compared to the SingleStep operating mode, an API function in this mode also overwrites the minimum cycle time of the OB1 cycle. When you define a minimum cycle time of 200 ms, the minimum distance between two cycle control points is 200 virtual milliseconds. The default mode is 100 ms.

The freeze state is terminated when the virtual controller issues an API command, performs the next step or changes to the default mode.

### API functions

- [GetOverwrittenMinimalCycleTime\\_ns\(\)](#) (Page 223)
- [SetOverwrittenMinimalCycleTime\\_ns\(\)](#) (Page 223)
- [OverwrittenMinimalCycleTime\\_ns { get; set; }](#) (Page 224)
- [RunNextCycle\(\)](#) (Page 225)

## See also

[Cycle control](#) (Page 221)

## 6.3.2 Synchronize simulation partner time-controlled

### Introduction

The TimespanSynchronized mode of the virtual controller provides one way to synchronize several simulation partners (clients).

### TimespanSynchronized operating mode

At least two simulation clients synchronize based on a virtual time period for this mode. A simulation client can be an instance of a virtual controller or an API client (an application that uses the Runtime API). The synchronization must be performed by a synchronization master.

The synchronization master signals a simulation client that it is his turn to run over certain time period. The time period is specified by the master in nanoseconds. The client then runs for the expected length of time before he goes into the freeze state at the next cycle control point. Before switching to the freeze state, the client sends the master the exact amount of time that he currently needed. Thereafter, the master signals the next client to catch up.

### API client as master

The API client as master signals each client when it should start. The master receives events from the clients when they occur.

An API client can only "time manage" instances of a virtual controller. The API client does not receive events from other API clients. It cannot send messages to other API clients.

### API functions

- Settings for cycle control (Page 221)
- StartProcessing() (Page 226)
- RunNextCycle() (Page 225)



## User interfaces (API)

### 7.1 Introduction

#### Components of the Simulation Runtime

The following components are relevant for handling the Simulation Runtime of PLCSIM Advanced:

- Runtime
  - Siemens.Simatic.Simulation.Runtime.Manager.exe
  - Siemens.Simatic.Simulation.Runtime.Instance.exe
- Libraries
  - Siemens.Simatic.Simulation.Runtime.Api.x86.dll
  - Siemens.Simatic.Simulation.Runtime.Api.x64.dll
  - SimulationRuntimeApi.h
- Documentation of interfaces with examples in native C++ and .NET (Managed Code)

#### Description

Runtime and libraries	Description
Siemens.Simatic.Simulation.Runtime.Manager.exe	A Windows process that runs in the background. Main component of Runtime that manages all other Runtime components.  The process is started automatically as soon as an application attempts to initialize the Runtime API. It is ended automatically as soon as there is no longer any application running that initialized the Runtime API.
Siemens.Simatic.Simulation.Runtime.Instance.exe	The process of the instance that loads a DLL of a virtual controller. Each virtual controller generates its own process.
Siemens.Simatic.Simulation.Runtime.Api.x86.dll Siemens.Simatic.Simulation.Runtime.Api.x64.dll	API libraries that must load an application to use the Simulation Runtime. The libraries contain interfaces for native code and managed code.  The "Runtime.Api.x86.dll" is loaded exclusively by 32-bit applications, and the "Runtime.Api.x64.dll" by 64-bit applications.
SimulationRuntimeApi.h	Header file that describes all data types that require a native C++ application to use the API library.

### External applications and Simulation Runtime

The following figure schematically presents the access of external applications to Simulation Runtime via the Runtime API. The Simulation Runtime Manager manages the Runtime instances. These load the libraries of the virtual controllers.

An external application can be other simulation software or a GUI, for example.

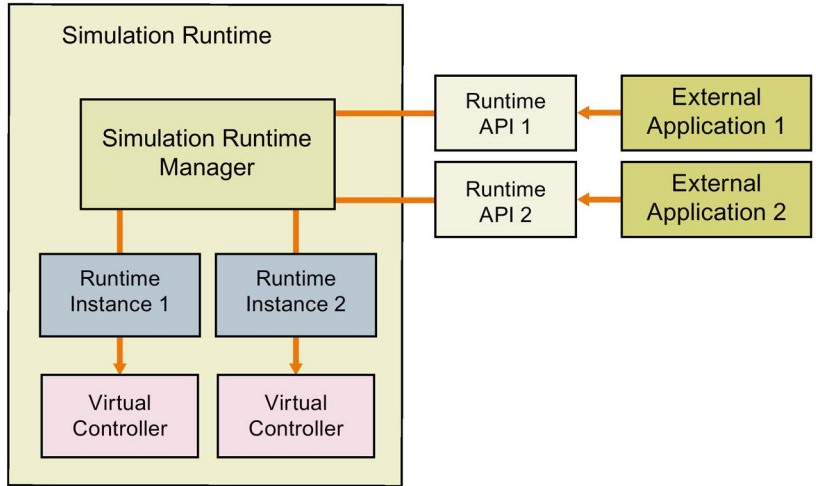
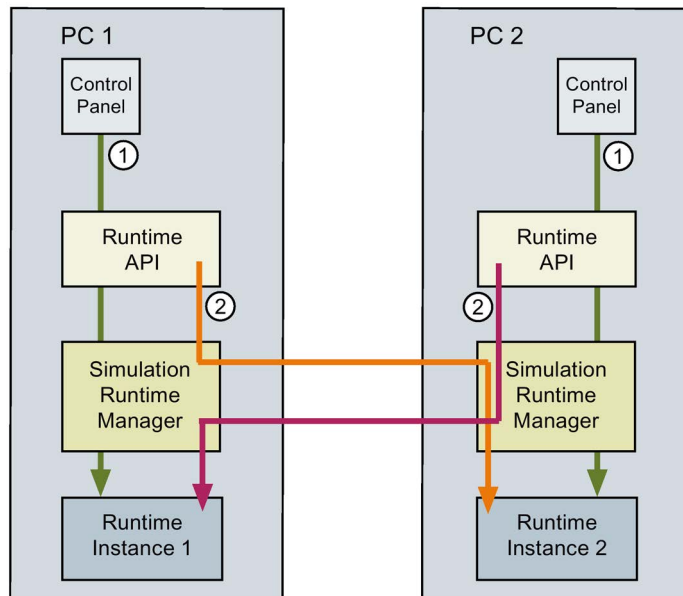


Figure 7-1 External applications and Simulation Runtime

## 7.1.1 Access to instances

### Access via the Control Panel and the API

You can access only one instance that is available locally on the PC via the Control Panel. It does not matter on which PC an instance was created and started. With distributed communication, the Runtime API accesses the instance of the other PCs via the Simulation Runtime Manager.



- ① Access to a local instance via the Control Panel
- ② Access to a remote instance on the Runtime API

Figure 7-2 Access to instances with distributed communication

### API functions

- Table 7-5 Overview of IRemoteRuntimeManager functions - Native C++ (Page 72)
- Table 7-9 Overview of IRemoteRuntimeManager functions - .NET (C#) (Page 75)
- Table 7-4 Overview of IInstances functions - Native C++ (Page 70)
- Table 7-8 Overview of IInstances functions - .NET (C#) (Page 74)

### See also

Overview of user interfaces for managed code (Page 73)

S7 PLCSIM Advanced Control Panel (Page 46)

### 7.1.2 User interfaces (API)

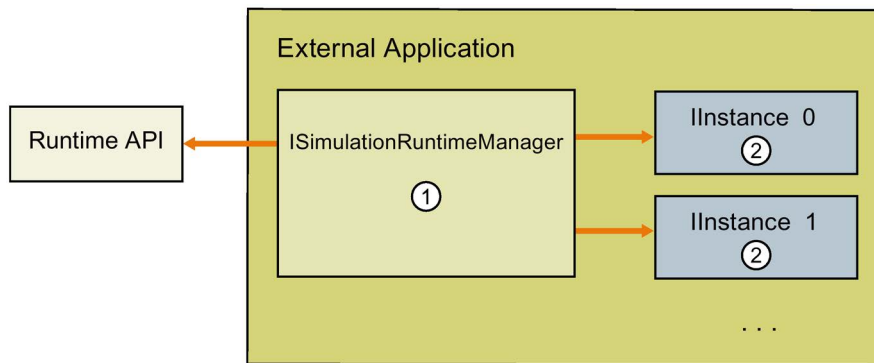
The user interfaces of Simulation Runtime include functions you use, for example, to create instances, to change the operating state of a virtual controller, or to exchange I/O data.

Simulation Runtime has the following user interfaces:

- ISimulationRuntimeManager
- IInstances
- IRemoteRuntimeManager

#### API and external applications

The Runtime API makes the interfaces available to an external application.



- ① **ISimulationRuntimeManager**  
Interface of the Runtime Manager. It is used to register new Runtime instances, to search through existing Runtime instances, and to receive an interface of a registered instance. Up to 16 instances can be registered in one Runtime Manager.
- ② **IInstances**  
Interface of a Runtime instance. It is used to change the operating state of a virtual controller and to exchange I/O data. Each instance has a unique name and an ID.

Figure 7-3 API and external applications

## Access to API functions and data types

Required functions and data types are available for native C++ and .NET (C#).

- Overview of user interfaces for native C++ (Page 69)
- Overview of data types for native C++ (Page 76)
- Overview of user interfaces for managed code (Page 73)
- Overview of data types for managed code (Page 77)

---

### Note

The list of tables in this manual gives you direct access to the description of the individual functions and data types.

---

## 7.1.3 Overview of user interfaces for native C++

### Initializing and shutting down API

The following table shows which functions are available for initializing and shutting down the API for native C++.

Table 7- 1 Overview of initializing and shutting down API - Native C++

Actions	Functions
Initialize API	InitializeApi (Page 78) RuntimeApiEntry_Initialize (Page 79)
Shut down API (Page 81)	DestroyInterface RuntimeApiEntry_DestroyInterface
Unregister DLL API (Page 84)	FreeApi ShutdownAndFreeApi

### Global functions

Table 7- 2 Overview of global functions - Native C++

Actions	Functions
Global functions (Page 86)	GetNameOfAreaSection() GetNameOfCPUType() GetNameOfCommunicationInterface() GetNameOfDataType() GetNameOfLEDMode() GetNameOfLEDType() GetNameOfOperatingMode() GetNameOfOperatingState() GetNameOfPrimitiveDataType() GetNameOfTagListDetails() GetNameOfErrorCode() GetNameOfRuntimeConfigChanged() GetNameOfInstanceConfigChanged()

### API ISimulationRuntimeManager

The following table shows which functions are available for the API ISimulationRuntimeManager.

Table 7-3 Overview of API ISimulationRuntimeManager functions - Native C++

Settings and information	Functions
Interface (Page 90)	GetVersion () IsInitialized () IsRuntimeManagerAvailable () Shutdown ()
Simulation Runtime instances (Page 93)	GetRegisteredInstancesCount () GetRegisteredInstanceInfoAt () RegisterInstance () RegisterCustomInstance () CreateInterface ()
Remote connections (Page 101)	OpenPort () ClosePort () GetPort () GetRemoteConnectionsCount () GetRemoteConnectionInfoAt () RemoteConnect ()
<b>Events</b>	
OnConfigurationChanged (Page 106)	RegisterOnConfigurationChangedCallback () UnregisterOnConfigurationChangedCallback () RegisterOnConfigurationChangedEvent () UnregisterOnConfigurationChangedEvent () WaitForOnConfigurationChangedEvent ()
OnRuntimeManagerLost (Page 110)	RegisterOnRuntimeManagerLostCallback () UnregisterOnRuntimeManagerLostCallback () RegisterOnRuntimeManagerLostEvent () UnregisterOnRuntimeManagerLostEvent () WaitForOnRuntimeManagerLostEvent ()

### API IInstances

The following table shows which functions are available for the API IInstances.

Table 7-4 Overview of IInstances functions - Native C++

Settings and information	Functions
Interface (Page 113)	GetID () GetName () GetCPUType () SetCPUType () GetCommunicationInterface () SetCommunicationInterface () GetInfo () UnregisterInstance ()
Controller (Page 119)	GetControllerName () GetControllerShortDesignation () GetControllerIPCount () GetControllerIP () GetControllerIPSuite4 () SetIPSuite () GetStoragePath () SetStoragePath () ArchiveStorage () RetrieveStorage ()

Settings and information	Functions
Operating state (Page 127)	GetOperatingState() PowerOn() PowerOff() MemoryReset() Run() Stop()
Tag list (Page 135)	UpdateTagList() GetTagListStatus() GetTagInfoCount() GetTagInfos() CreateConfigurationFile()
I/O access via address - Reading (Page 141)	GetAreaSize() ReadBit() ReadByte() ReadBytes() ReadSignals()
I/O access via address - Writing (Page 150)	WriteBit() WriteByte() WriteBytes() WriteSignals()
I/O access via tag name - Reading (Page 158)	Read() ReadBool() ReadChar(), ReadWChar() ReadDouble() ReadFloat() ReadInt8(), ReadInt16(), ReadInt32(), ReadInt64() ReadUInt8(), ReadUInt16(), ReadUInt32(), ReadUInt64() ReadSignals()
I/O access via tag name - Writing (Page 188)	Write() WriteBool() WriteChar(), WriteWChar() WriteDouble() WriteFloat() WriteInt8(), WriteInt16(), WriteInt32(), WriteInt64(), WriteUInt8(), WriteUInt16(), WriteUInt32(), WriteUInt64() WriteSignals()
Virtual time (Page 218)	GetSystemTime() SetSystemTime() GetScaleFactor() SetScaleFactor()
Cycle control (Page 221)	GetOperatingMode() SetOperatingMode() SetAlwaysSendOnEndOfCycleEnabled() IsAlwaysSendOnEndOfCycleEnabled() GetOverwrittenMinimalCycleTime_ns() SetOverwrittenMinimalCycleTime_ns() RunNextCycle() StartProcessing()
<b>Events</b>	
OnOperatingStateChanged (Page 228)	RegisterOnOperatingStateChangedCallback() UnregisterOnOperatingStateChangedCallback() RegisterOnOperatingStateChangedEvent() UnregisterOnOperatingStateChangedEvent() WaitForOnOperatingStateChangedEvent()
OnEndOfCycle (Page 232)	RegisterOnEndOfCycleCallback() UnregisterOnEndOfCycleCallback() RegisterOnEndOfCycleEvent() UnregisterOnEndOfCycleEvent() WaitForOnEndOfCycleEvent()
OnConfigurationChanging (Page 234)	RegisterOnConfigurationChangingCallback() UnregisterOnConfigurationChangingCallback() RegisterOnConfigurationChangingEvent() UnregisterOnConfigurationChangingEvent() WaitForOnConfigurationChangingEvent()

Settings and information	Functions
OnConfigurationChanged (Page 237)	RegisterOnConfigurationChangedCallback () UnregisterOnConfigurationChangedCallback () RegisterOnConfigurationChangedEvent () UnregisterOnConfigurationChangedEvent () WaitForOnConfigurationChangedEvent ()
OnLedChanged (Page 239)	RegisterOnLedChangedCallback () UnregisterOnLedChangedCallback () RegisterOnLedChangedEvent () UnregisterOnLedChangedEvent () WaitForOnLedChangedEvent ()

### API IRemoteRuntimeManager

The following table shows which functions are available for the API IRemoteRuntimeManager (remote connections):

Table 7- 5 Overview of IRemoteRuntimeManager functions - Native C++

Settings and information	Functions
Interface (Page 242)	GetVersion () GetIP () GetPort () GetRemoteComputerName () Disconnect ()
Simulation Runtime instances (Page 246)	GetRegisteredInstancesCount () GetRegisteredInstanceInfoAt () RegisterInstance () RegisterCustomInstance () CreateInterface ()
<b>Events</b>	
OnConnectionLost (Page 254)	RegisterOnConnectionLostCallback () UnregisterOnConnectionLostCallback () RegisterOnConnectionLostEvent () UnregisterOnConnectionLostEvent () WaitForOnConnectionLostEvent ()



## 7.1.4 Overview of user interfaces for managed code

### Initializing and shutting down API

Table 7- 6 Overview of initializing and shutting down API - .NET (C#)

Actions	Functions
Initialize API (Page 80)	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeManager
Shut down API (Page 86)	

### API ISimulationRuntimeManager

The following table shows which functions are available for the API ISimulationRuntimeManager.

Table 7- 7 Overview of ISimulationRuntimeManager functions - .NET (C#)

Settings and information	Functions
Interface (Page 90)	Version { get; } IsInitialized { get; } IsRuntimeManagerAvailable { get; } Shutdown()
Simulation Runtime instances (Page 93)	RegisterInstanceInfo { get; } RegisterInstance() RegisterCustomInstance() CreateInterface()
Remote connections (Page 101)	OpenPort() ClosePort() Port { get; } RemoteConnectionInfo { get; } RemoteConnect()
<b>Events</b>	
OnConfigurationChanged (Page 106)	OnConfigurationChanged RegisterOnConfigurationChangedEvent() UnregisterOnConfigurationChangedEvent() WaitForOnConfigurationChangedEvent()
OnRuntimeManagerLost (Page 110)	OnRuntimeManagerLost() RegisterOnRuntimeManagerLostEvent() UnregisterOnRuntimeManagerLostEvent() WaitForOnRuntimeManagerLostEvent()

## API Instances

The following table shows which functions are available for the API Instances.

Table 7- 8 Overview of Instances functions - .NET (C#)

Settings and information	Functions
Interface (Page 113)	Dispose () ID { get; } Name { get; } CPUType { get; set; } CommunicationInterface { get; } Info { get; } UnregisterInstance()
Controller - Information and settings (Page 119)	ControllerName { get; } ControllerShortDesignation { get; } ControllerIPSuite4 { get; } SetIPSuite() StoragePath { get; set; } ArchiveStorage() RetrieveStorage()
Operating state (Page 127)	OperatingState { get; } PowerOn() PowerOff() MemoryReset() Run() Stop()
Tag list (Page 135)	UpdateTagList() GetTagListStatus() TagInfos { get; } CreateConfigurationFile()
I/O access via address - Reading (Page 141)	InputArea   MarkerArea   OutputArea { get; } AreaSize { get; } ReadBit() ReadByte() ReadBytes() ReadSignals()
I/O access via address - Writing (Page 150)	WriteBit() WriteByte() WriteBytes() WriteSignals()
I/O access via tag name - Reading (Page 158)	Read() ReadBool() ReadChar(), ReadWChar() ReadDouble() ReadFloat() ReadInt8(), ReadInt16(), ReadInt32(), ReadInt64() ReadUInt8(), ReadUInt16(), ReadUInt32(), ReadUInt64() ReadSignals()
I/O access via tag name - Writing (Page 188)	Write() WriteBool() WriteChar(), WriteWChar() WriteDouble() WriteFloat() WriteInt8(), WriteInt16(), WriteInt32(), WriteInt64(), WriteUInt8(), WriteUInt16(), WriteUInt32(), WriteUInt64() WriteSignals()
Virtual time (Page 218)	SystemTime { get; set; } ScaleFactor { get; set; }
Cycle control (Page 221)	OperatingMode { get; set; } IsAlwaysSendOnEndOfCycleEnabled { get; set; } OverwrittenMinimalCycleTime_ns { get; set; } RunNextCycle() StartProcessing()

Settings and information	Functions
<b>Events</b>	
OnOperatingStateChanged (Page 228)	OnOperatingStateChanged RegisterOnOperatingStateChangedEvent () UnregisterOnOperatingStateChangedEvent () WaitForOnOperatingStateChangedEvent ()
OnEndOfCycle (Page 232)	OnEndOfCycle RegisterOnEndOfCycleEvent () UnregisterOnEndOfCycleEvent () WaitForOnEndOfCycleEvent ()
OnConfigurationChanging (Page 234)	OnConfigurationChanging RegisterOnConfigurationChangingEvent () UnregisterOnConfigurationChangingEvent () WaitForOnConfigurationChangingEvent ()
OnConfigurationChanged (Page 106)	OnConfigurationChanged RegisterOnConfigurationChangedEvent () UnregisterOnConfigurationChangedEvent () WaitForOnConfigurationChangedEvent ()
OnLedChanged (Page 239)	OnLedChanged RegisterOnLedChangedEvent () UnregisterOnLedChangedEvent () WaitForOnLedChangedEvent ()

## API IRemoteRuntimeManager

The following table shows which functions are available for the API IRemoteRuntimeManager.

Table 7- 9 Overview of IRemoteRuntimeManager functions - .NET (C#)

Settings and information	Functions
Interface (Page 242)	Dispose () Version { get; } IP { get; } Port { get; } RemoteComputerName { get; } Disconnect ()
Simulation Runtime instances (Page 93)	RegisterInstanceInfo { get; } RegisterInstance () RegisterCustomInstance () CreateInterface ()
<b>Events</b>	
OnConnectionLost() (Page 254)	OnConnectionLost () RegisterOnConnectionLostEvent () UnregisterOnConnectionLostEvent () WaitForOnConnectionLostEvent ()

### 7.1.5 Overview of data types for native C++

The following table shows which data types are available for the simulation in Runtime Manager.

Table 7- 10 Overview of data types - Native C++

Data type	
DLL import functions (Page 258)	ApiEntry_Initialize ApiEntry_DestroyInterface
Event callback functions (Page 259)	EventCallback_VOID EventCallback_II_SREC_ST EventCallback_II_SREC_ST_SROS_SROS EventCallback_II_SREC_ST_SRLT_SRLM EventCallback_II_SREC_ST_INT64_UINT32 EventCallback_IRRTM EventCallback_SRCC_UINT32_UINT32_INT32 EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32
Definitions and constants (Page 271)	
Unions (Page 272)	UIP UDataValue
Structures (Page 274)	SDataValue SDataValueByAddress SDataValueByName SConnectionInfo SInstanceInfo SDimension STagInfo SIP SIPSuite4
Enumerations (Page 284)	ERuntimeErrorCode EArea EOperatingState EOperatingMode ECPUType ECommunicationInterface ELEDType ELEDMode EPrimitiveDataTypes EDataType ETagListDetails ERuntimeConfigChanged EInstanceConfigChanged

## 7.1.6 Overview of data types for managed code

The following table shows which data types are available for the simulation in Runtime Manager.

Table 7- 11 Overview of data types - .NET (C#)

Data type	
Delegate definitions (Page 265) - Event handler methods	Delegate_Void Delegate_II_EREC_DT Delegate_II_EREC_DT_EOS_EOS Delegate_II_EREC_DT_ELT_ELM Delegate_II_EREC_DT_INT64_UINT32 Delegate_IRRTM Delegate_SRCC_UINT32_UINT32_INT32 Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32
Definitions and constants (Page 271)	
Structures (Page 274)	SDataValue SDataValueByAddress SDataValueByName SConnectionInfo SInstanceInfo SDimension STagInfo SIP SIPSuite4
Enumerations (Page 284)	ERuntimeErrorCode EArea EOperatingState EOperatingMode ECPUType ECommunicationInterface ELEDType ELEDMode EPrimitiveDataTypes EDataType ETagListDetails ERuntimeConfigChanged EInstanceConfigChanged

## 7.2 Initialize API

### 7.2.1 Native C++

#### 7.2.1.1 InitializeApi()

#### Description

The `InitializeApi` function loads the API library (DLL) and initializes the API. The function loads the version of the DLL that is compatible with the architecture of your application. The DLL is loaded from the Startup directory of each application that calls this function or from the directory that provides the path parameter.

The function returns an interface to the Simulation Runtime Manager. Use this interface to create a new instance of the virtual controller or to obtain access to an existing instance.

Table 7- 12 InitializeApi() - Native C++

Syntax	<pre>ERuntimeErrorCode InitializeApi(     ISimulationRuntimeManager** out_SimulationRuntimeManagerInterface ); ERuntimeErrorCode InitializeApi(     WCHAR* in_SimulationRuntimeApiDllPath,     ISimulationRuntimeManager** in-     out_SimulationRuntimeManagerInterface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>ISimulationRuntimeManager** out_SimulationRuntimeManagerInterface: Pointer to a Runtime Manager interface pointer. The pointer must be initialized with <code>ZERO</code>. The interface is created within the function. See Data types (Page 257).</li> <li>WCHAR* in_SimulationRuntimeApiDllPath: The path to the Runtime API library (DLL).</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_WRONG_ARGUMENT	The pointer to the Runtime Manager interface does not equal <code>ZERO</code> .
	SREC_WRONG_VERSION	The required version of the interface is incompatible with the version used to compile the API.
	SREC_CONNECTION_ERROR	Unable to establish a connection to the Runtime Manager.
	SREC_ERROR_LOADING_DLL	The API library (DLL) cannot be loaded.

Example C++	<pre>// Include The Headerfile Of The API #include "SimulationRuntimeApi.h"  // Prepare The Variables ERuntimeErrorCode result = SREC_INVALID_ERROR_CODE; ISimulationRuntimeManager* api = ZERO;  // Initialize The API And Get The RuntimeManager Interface result = <b>InitializeApi</b>(&amp;api);</pre>
-------------	---

**Note**

If you no longer require the interface, delete it.

See DestroyInterface() (Page 82).

### 7.2.1.2 RuntimeApiEntry\_Initialize

**Description**

Use the `RuntimeApiEntry_Initialize` function only if the API library (DLL) is to be loaded from a different directory than the Startup directory of the application that calls this function.

When the API is initialized, the API library is first loaded and the `Initialize` function is then imported and called.

The function returns an interface to the Simulation Runtime Manager. Use this interface to create a new instance of the virtual controller or to obtain access to an existing instance.

Table 7- 13 RuntimeApiEntry\_Initialize - Native C++

Syntax	<pre>_declspec(dllexport) ERuntimeErrorCode RuntimeApiEntry_Initialize(     ISimulationRuntimeManager** out_SimulationRuntimeManagerInterface,     UINT32 in_InterfaceVersion );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>ISimulationRuntimeManager**</code> <code>out_SimulationRuntimeManagerInterface:</code> Pointer to a Runtime Manager interface pointer. The pointer must be initialized with <code>NULL</code>. The interface is created within the function. See Data types (Page 257).</li> <li>• <code>UINT32 in_InterfaceVersion:</code> Version of the API interface to be downloaded: <code>DAPI_DLL_INTERFACE_VERSION</code>.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_WRONG_ARGUMENT</code>	The pointer to the Runtime Manager interface does not equal <code>NULL</code> .
	<code>SREC_WRONG_VERSION</code>	The required version of the interface is incompatible with the version used to compile the API.

7.2 Initialize API

	SREC_CONNECTION_ERROR	Unable to establish a connection to the Runtime Manager.
Example C++	<pre>// Include The Headerfile Of The API #include "SimulationRuntimeApi.h"  // Prepare The Variables ERuntimeErrorCode result = SREC_INVALID_ERROR_CODE; HMODULE dllHandle = NULL; ApiEntry_Initialize Initialize = NULL; ISimulationRuntimeManager* api = NULL;  // Load The DLL And Import The "Initialize" Function (using the Win32 API) dllHandle = LoadLibrary(DAPI_DLL_NAME_X86); if ( dllHandle != NULL ) {     Initialize = (ApiEntry_Initialize)GetProcAddress(dllHandle, DAPI_ENTRY_INITIALIZE); }  // Initialize The API And Get The RuntimeManager Interface if ( Initialize != NULL ) {     result = <b>Initialize</b>(&amp;api, DAPI_DLL_INTERFACE_VERSION); } </pre>	

**Note**

If you no longer require the interface, delete it.  
See DestroyInterface() (Page 82).

7.2.2 .NET (C#)

7.2.2.1 Initialize

**Description**

The entry point to the API is the static class  
Siemens.Simatic.Simulation.Runtime.SimulationRuntimeManager.  
The API is initialized when a function of this class is used the first time.

Table 7- 14 Initialize - .NET (C#)

Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationInitializationException	
	Runtime error code	Condition
	ERuntimeError-Code.ConnectionError	Unable to establish a connection to the Runtime Manager.



## 7.3 Shut down API

### 7.3.1 Native C++

#### Basic procedure for deleting the user interfaces

To delete all user interfaces, generally follow these steps:

1. Delete the interfaces `IInstances` and `IRemoteRuntimeManager`.
2. Call the `Shutdown()` function of the `ISimulationRuntimeManager` interface.
3. Delete the `ISimulationRuntimeManager` interface.
4. Unload the API library (DLL) with the Win32 API-Funktion `FreeLibrary()`.

#### Deleting the user interfaces via functions

Deleting the user interfaces is also possible via functions.

If the API was initialized using the `InitializeApi()` function, you delete the user interfaces using the following functions:

- `FreeApi()` (Page 84)
- `ShutdownAndFreeApi()` (Page 85)

### 7.3.1.1 DestroyInterface()

#### Description

A function pointer to the `RuntimeApiEntry_DestroyInterface` function. The function pointer `DestroyInterface()` is only valid if the `InitializeApi` function has been successfully called.

The function unloads the memory of an `ISimulationRuntimeManager`, `IRemoteRuntimeManager` or `IInstance` interface.

Table 7- 15 DestroyInterface() - Native C++

Syntax	<code>ERuntimeErrorCode DestroyInterface( _IBaseInterface* in_Interface );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>_IBaseInterface* in_Interface:</code> The interface to be deleted.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_WRONG_ARGUMENT</code>	The pointer to the interface is <code>NULL</code> .
Example C++	<pre>// Include The Headerfile Of The API #include "SimulationRuntimeApi.h"  // The Interfaces ERuntimeErrorCode result; ISimulationRuntimeManager* api = NULL; IInstance* instance = NULL;  // Init the DLL and create an instance result = InitializeApi(&amp;api); result = api-&gt;RegisterInstance(&amp;instance);  // Destroy Instance Interfaces result = <b>DestroyInterface</b>(instance); instance = NULL;</pre>	

### 7.3.1.2 RuntimeApiEntry\_DestroyInterface

#### Description

Use the `RuntimeApiEntry_DestroyInterface` function only if the API library (DLL) is to be loaded from a different directory than the Startup directory of the application that calls this function.

If the API was initialized using the `InitializeApi` function, you select the `DestroyInterface()` (Page 82) function.

The function unloads the memory of an `ISimulationRuntimeManager`, `IRemoteRuntimeManager` or `IInstance` interface.

Table 7- 16 RuntimeApiEntry\_DestroyInterface() - Native C++

Syntax	<pre> _declspec(dllexport) HRESULT RuntimeA- piEntry_DestroyInterface(     IBaseInterface* in_Interface ); </pre>	
Parameters	<ul style="list-style-type: none"> <li><code>IBaseInterface* in_Interface</code>: The interface to be deleted.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_WRONG_ARGUMENT</code>	The pointer to the interface is <code>NULL</code> .
Example C++	<pre> // Include The Headerfile Of The API #include "SimulationRuntimeApi.h"  // Prepare The Variables ERuntimeErrorCode result = SREC_INVALID_ERROR_CODE; HMODULE dllHandle = NULL; ApiEntry_DestroyInterface Destroy = NULL; IInstance* instance = NULL;  // Load The DLL And Import The "DestroyInterface" Function (using the Win32 API) dllHandle = LoadLibraryA(DAPI_DLL_NAME_X86); if ( dllHandle != NULL ) {     Destroy = (ApiEntry_DestroyInterface)GetProcAddress(dllHandle, DAPI_ENTRY_DESTROY_INTERFACE); } ... // Frees the memory of an IInstance interface result = Destroy(instance); </pre>	

### 7.3.1.3 FreeApi()

#### Description

The `FreeApi()` function unloads the library of the Runtime API.

This function can only be called after the successful call of the `InitializeApi` function. If the `InitializeApi` function was not called, the library must be unloaded using the Win32 API function `FreeLibrary()`.

Table 7- 17 FreeApi() - Native C++

Syntax	<code>ERuntimeErrorCode FreeApi();</code>	
Parameters	None	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_API_NOT_INITIALIZED</code>	The <code>InitializeApi</code> function was not called successfully.
Example C++	<pre>// Include The Headerfile Of The API #include "SimulationRuntimeApi.h"  // The Interfaces ERuntimeErrorCode result; ISimulationRuntimeManager* api = NULL; IInstance* instance = NULL;  // Init the API result = InitializeApi(&amp;api);  ...  // Shutdown The API api-&gt;Shutdown(); result = DestroyInterface(api); api = NULL; result = <b>FreeApi()</b>;</pre>	

### 7.3.1.4 ShutdownAndFreeApi()

#### Description

The `ShutdownAndFreeApi()` function shuts down the Runtime API, deletes the `IRuntimeManager` interface and unloads the library of the Runtime API.

This function can only be called after the successful call of the `InitializeApi` function. If the `InitializeApi` function was not called, the library must be unloaded using the Win32 API-Funktion `FreeLibrary()`.

Table 7- 18 ShutdownAndFreeApi() - Native C++

Syntax	<code>ERuntimeErrorCode ShutdownAndFreeApi( ISimulationRuntimeManager* in_SimulationRuntimeManagerInterface );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>ISimulationRuntimeManager* in_SimulationRuntimeManagerInterface:</code> The interface of the Runtime Manager to be deleted.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_API_NOT_INITIALIZED</code>	The <code>InitializeApi</code> function was not called successfully.
	<code>SREC_WRONG_ARGUMENT</code>	The pointer to the interface is <code>NULL</code> .
Example C++	<pre>// Include The Headerfile Of The API #include "SimulationRuntimeApi.h"  // The Interfaces ERuntimeErrorCode result; ISimulationRuntimeManager* api = NULL; IInstance* instance = NULL;  // Init the API result = InitializeApi(&amp;api);  ...  // Shutdown The API result = ShutdownAndFreeApi(api); api = NULL;</pre>	

## 7.3.2 .NET (C#)

### 7.3.2.1 Shut down API

You can end the .NET components of the API in the following ways:

- By a clearing operation using the .NET Garbage Collector.
- For the IInstance and IRemoteRuntimeManager interfaces, by calling the Dispose (Page 113) function.

### Manually clearing the API

To manually clear the API, follow these steps:

1. Delete all interfaces. Interfaces - Information and settings (Page 113)
2. Call the Shutdown() (Page 90) function of the ISimulationRuntimeManager interface.

## 7.4 Global functions (Native C++)

### GetNameOfAreaSection()

Returns the name of the enumeration entry.

Table 7- 19 GetNameOfAreaSection() - Native C++

Syntax	<code>const WCHAR* GetNameOfAreaSection( EArea in_AreaSection );</code>
Parameters	<ul style="list-style-type: none"> <li>• EArea in_AreaSection:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

### GetNameOfCPUType()

Returns the name of the enumeration entry.

Table 7- 20 GetNameOfCPUType() - Native C++

Syntax	<code>const WCHAR* GetNameOfCPUType( ECPUType in_CPUType );</code>
Parameters	<ul style="list-style-type: none"> <li>• ECPUType in_CPUType:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

**GetNameOfCommunicationInterface()**

Returns the name of the enumeration entry.

Table 7- 21 GetNameOfCommunicationInterface() - Native C++

Syntax	<code>const WCHAR* GetNameOfCommunicationInterface( ECommunicationInterface in_CommunicationInterface );</code>
Parameters	<ul style="list-style-type: none"> <li>ECommunicationInterface in_CommunicationInterface:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

**GetNameOfDataType()**

Returns the name of the enumeration entry.

Table 7- 22 GetNameOfDataType() - Native C++

Syntax	<code>const WCHAR* GetNameOfDataType( EDatatype in_DataType );</code>
Parameters	<ul style="list-style-type: none"> <li>EDatatype in_DataType:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

**GetNameOfErrorCode()**

Returns the name of the enumeration entry.

Table 7- 23 GetNameOfErrorCode() - Native C++

Syntax	<code>const WCHAR* GetNameOfErrorCode( ERuntimeErrorCode in_ErrorCode );</code>
Parameters	<ul style="list-style-type: none"> <li>ERuntimeErrorCode in_ErrorCode:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

**GetNameOfLEDMode()**

Returns the name of the enumeration entry.

Table 7- 24 GetNameOfLEDMode() - Native C++

Syntax	<code>const WCHAR* GetNameOfLEDMode( ELEDMode in_LEDMode );</code>
Parameters	<ul style="list-style-type: none"> <li>ELEDMode in_LEDMode:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

### GetNameOfLEDType()

Returns the name of the enumeration entry.

Table 7- 25 GetNameOfLEDType() - Native C++

Syntax	<code>const WCHAR* GetNameOfLEDType( ELEDType in_LEDType );</code>
Parameters	<ul style="list-style-type: none"> <li>ELEDType in_LEDType:  Enumeration entry.</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

### GetNameOfOperatingMode()

Returns the name of the enumeration entry.

Table 7- 26 GetNameOfOperatingMode() - Native C++

Syntax	<code>const WCHAR* GetNameOfOperatingMode( EOperatingMode in_OperatingMode );</code>
Parameters	<ul style="list-style-type: none"> <li>EOperatingMode in_OperatingMode:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

### GetNameOfErrorCode()

Returns the name of the enumeration entry.

Table 7- 27 GetNameOfErrorCode() - Native C++

Syntax	<code>const WCHAR* GetNameOfErrorCode( ERuntimeErrorCode in_ErrorCode );</code>
Parameters	<ul style="list-style-type: none"> <li>ERuntimeErrorCode in_ErrorCode:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

### GetNameOfOperatingState

Returns the name of the enumeration entry.

Table 7- 28 GetNameOfOperatingState() - Native C++

Syntax	<code>const WCHAR* GetNameOfOperatingState( EOperatingState in_OperatingState );</code>
Parameters	<ul style="list-style-type: none"> <li>EOperatingState in_OperatingState:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry



## GetNameOfPrimitiveDataType

Returns the name of the enumeration entry.

Table 7- 29 GetNameOfPrimitiveDataType() - Native C++

Syntax	<code>const WCHAR* GetNameOfPrimitiveDataType( EPrimitiveDataType in_DataType );</code>
Parameters	<ul style="list-style-type: none"> <li>EPrimitiveDataType in_DataType:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

## GetNameOfTagListDetails

Returns the name of the enumeration entry.

Table 7- 30 GetNameOfTagListDetails() - Native C++

Syntax	<code>const WCHAR* GetNameOfTagListDetails( ETagListDetails in_TagListDetails );</code>
Parameters	<ul style="list-style-type: none"> <li>ETagListDetails in_TagListDetails:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

## GetNameOfRuntimeConfigChanged()

Returns the name of the enumeration entry.

Table 7- 31 GetNameOfRuntimeConfigChanged() - Native C++

Syntax	<code>const WCHAR* GetNameOfRuntimeConfigChanged( ERuntimeConfigChanged in_RuntimeConfigChanged);</code>
Parameters	<ul style="list-style-type: none"> <li>ERuntimeConfigChanged in_RuntimeConfigChanged:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

## GetNameOfInstanceConfigChanged()

Returns the name of the enumeration entry.

Table 7- 32 GetNameOfInstanceConfigChanged() - Native C++

Syntax	<code>const WCHAR* GetNameOfInstanceConfigChanged( EInstanceCon- figChanged in_InstanceConfigChanged);</code>
Parameters	<ul style="list-style-type: none"> <li>EInstanceConfigChanged in_InstanceConfigChanged:  Enumeration entry</li> </ul>
Return values	<code>const WCHAR*</code> : Name of the enumeration entry

**See also**

EPrimitiveDataType (Page 294)

EDatatype (Page 296)

## 7.5 API ISimulationRuntimeManager

### 7.5.1 Interfaces - Information and settings

#### GetVersion() / Version { get; }

Returns the version of Runtime Manager. If the function fails, version 0.0 is returned.

Table 7- 33 GetVersion() - Native C++

Syntax	<code>UINT32 GetVersion();</code>
Parameters	None
Return values	<code>UINT32: Runtime Manager Version (HIWORD = Major, LOWORD = Minor)</code>

Table 7- 34 Version { get; } - .NET (C#)

Syntax	<code>UInt32 Version { get; }</code>
Parameters	None
Return values	<code>UInt32: Runtime Manager Version (HIWORD = Major, LOWORD = Minor)</code>

**IsInitialized() / IsInitialized { get; }**

Returns a value that indicates whether the API was successfully initialized.

Table 7- 35 IsInitialized() - Native C++

Syntax	<code>bool IsInitialized();</code>
Parameters	None
Return values	<ul style="list-style-type: none"> <li>• <code>false</code>: If the API was not initialized.</li> <li>• <code>true</code>: If the API was initialized.</li> </ul>

Table 7- 36 IsInitialized { get; } - .NET (C#)

Syntax	<code>bool IsInitialized { get; }</code>
Parameters	None
Return values	<ul style="list-style-type: none"> <li>• <code>false</code>: If the API was not initialized.</li> <li>• <code>true</code>: If the API was initialized.</li> </ul>

**IsRuntimeManagerAvailable() / IsRuntimeManagerAvailable { get; }**

The function returns `false` when the connection to Runtime Manager is interrupted. This happens only when the Runtime Manager process is closed.

Subscribe to the `OnRuntimeManagerLost()` event to find out whether the connection is interrupted. See Events (Page 106).

Table 7- 37 IsRuntimeManagerAvailable() - Native C++

Syntax	<code>bool IsRuntimeManagerAvailable();</code>
Parameters	None
Return values	<ul style="list-style-type: none"> <li>• <code>false</code>: If the connection is interrupted.</li> <li>• <code>true</code>: If the connection is active.</li> </ul>

Table 7- 38 IsRuntimeManagerAvailable { get; } - .NET (C#)

Syntax	<code>bool IsRuntimeManagerAvailable{ get; }</code>
Parameters	None
Return values	<ul style="list-style-type: none"> <li>• <code>false</code>: If the connection is interrupted.</li> <li>• <code>true</code>: If the connection is active.</li> </ul>

### Shutdown()

Ends communication with Runtime Manager and clears the interfaces.

Call this function in the following cases:

- Immediately before the API library (DLL) is unregistered (native C++).
- When your application is no longer using Runtime Manager.

Table 7- 39 Shutdown() - Native C++

Syntax	ERuntimeErrorCode Shutdown()	
Parameters	None	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.

Table 7- 40 Shutdown() - .NET (C#)

Syntax	void Shutdown()
Parameters	None
Return values	None

## 7.5.2 Simulation Runtime instances

### GetRegisteredInstancesCount()

Returns the number of instances that are registered in Runtime Manager. If the function fails, the return value is 0.

Table 7- 41 GetRegisteredInstancesCount() - Native C++

Syntax	UINT32 GetRegisteredInstancesCount();
Parameters	None
Return values	UINT32: Number of available instances.

### GetRegisteredInstanceInfoAt()

Returns information about an already registered instance. You can use the ID or name to create an interface of this instance, see `CreateInterface()`.

Table 7- 42 GetRegisteredInstanceInfoAt() - Native C++

Syntax	ERuntimeErrorCode GetRegisteredInstanceInfoAt (UINT32 in_Index, SInstanceInfo* out_InstanceInfo);	
Parameters	<ul style="list-style-type: none"> <li>• <code>UINT32 in_Index</code>: Index of the created instance from which you want to receive the information. The index must be less than the value you receive when you call <code>GetRegisteredInstanceCount()</code>.</li> <li>• <code>SInstanceInfo* out_InstanceInfo</code>: The information with name and ID of the instance. See Data types (Page 271).</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INDEX_OUT_OF_RANGE	There is no instance information for this index.

**RegisteredInstanceInfo { get; }**

Returns information about all already registered instances. Use the ID or name of this instance to create an interface of this instance, see `CreateInterface()`.

Table 7- 43 RegisteredInstanceInfo { get; } - .NET (C#)

Syntax	<code>SInstanceInfo[] RegisteredInstanceInfo { get; }</code>	
Parameters	None	
Return values	<code>SInstanceInfo[]</code> : An array of information about all registered instances.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.

## RegisterInstance()

Registers a new instance of a virtual controller in Runtime Manager. Creates and returns an interface of this instance.

Table 7- 44 RegisterInstance() - Native C++

Syntax	<pre>ERuntimeErrorCode RegisterInstance(     IInstance** out_InstanceInterface ); ERuntimeErrorCode RegisterInstance(     WCHAR* in_InstanceName,     IInstance** out_InstanceInterface ); ERuntimeErrorCode RegisterInstance(     ECPUType in_CPUType,     IInstance** out_InstanceInterface ); ERuntimeErrorCode RegisterInstance(     ECPUType in_CPUType,     WCHAR* in_InstanceName,     IInstance** out_InstanceInterface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• ECPUType in_CPUType: <p>Defines which CPU type is simulated at the start of the instance. The default setting is "SRCT_1500_Unspecified".</p> <p>When a different CPU type is loaded via STEP 7 or from the Virtual Memory Card, this CPU type applies.</p> </li> <li>• WCHAR* in_InstanceName: <p>Name to be assigned to the instance. Every instance must have a unique name. If no name is assigned when registering a new instance, the instance is given the name "Instance_#" (# is the ID of the instance). If this name already exists, the name "Instance_#.#" is used, in which the second # is a counter that is incremented until the name is unique. The length of the name must be less than DINSTANCE_NAME_LENGTH. See Data types (Page 271).</p> </li> <li>• IInstance** out_InstanceInterface: <p>Pointer to a Simulation Runtime interface pointer. The pointer must be initialized with ZERO. The interface is created within the function.</p> </li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_TIMEOUT	The function does not return on time.
	SREC_WRONG_ARGUMENT	The name or the IInstance pointer is invalid.
	SREC_LIMIT_REACHED	There are already 16 instances registered in Runtime Manager.
SREC_ALREADY_EXISTS	An instance with this name already exists.	
Example C++	<pre>ISimulationRuntimeManager * api = ZERO; ERuntimeErrorCode result = Initialize(&amp;api);  // Example: How To Create And Register An Instance IInstance* psa = ZERO; if (result == SREC_OK) {     result = api-&gt;RegisterInstance(&amp;psa); }</pre>	

**Note**

**Native C++**

If you no longer require the interface, delete it.

See DestroyInterface() (Page 82).

Table 7- 45 RegisterInstance() - .NET (C#)

Syntax	<pre> IInstance RegisterInstance(); IInstance RegisterInstance(     string in_InstanceName ); IInstance RegisterInstance(     ECPUType in_CPUType ); IInstance RegisterInstance(     ECPUType in_CPUType     string in_InstanceName );                 </pre>	
Parameters	<ul style="list-style-type: none"> <li> <b>ECPUType in_CPUType:</b>                      Defines which CPU type is simulated at the start of the instance. The default setting is "ECPUType.Unspecified".                       When a different CPU type is loaded via STEP 7 or from the Virtual Memory Card, this CPU type applies.                 </li> <li> <b>string in_InstanceName:</b>                      Name to be assigned to the instance. Every instance must have a unique name. If no name is assigned when registering a new instance, the instance is given the name "Instance_#" (# is the ID of the instance). If this name already exists, the name "Instance_#. #" is used, in which the second # is a counter that is incremented until the name is unique. The length of the name must be less than <code>DINSTANCE_NAME_LENGTH</code>. See Data types (Page 271).                 </li> </ul>	
Return values	If the function is successful, an interface of a virtual controller, otherwise a null pointer.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeErrorCode.WrongArgument</code>	The name is invalid.
	<code>ERuntimeErrorCode.LimitReached</code>	There are already 16 instances registered in Runtime Manager.
<code>ERuntimeErrorCode.AlreadyExists</code>	An instance with this name already exists.	



**RegisterCustomInstance()**

Registers a new instance of a virtual controller in Runtime Manager. Creates and returns an interface of this instance.

Table 7- 46 RegisterCustomInstance() - Native C++

Syntax	<pre>ERuntimeErrorCode RegisterCustomInstance(     WCHAR* in_VplcDll,     IInstance** out_InstanceInterface ); ERuntimeErrorCode RegisterCustomInstance(     WCHAR* in_VplcDll,     WCHAR* in_InstanceName,     IInstance** out_InstanceInterface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_VplcDll: The complete path to the DLL of the virtual controller that Siemens.Simatic.Simulation.Runtime.Instance.exe loads at PowerOn.</li> <li>WCHAR* in_InstanceName: Name to be assigned to the instance. Every instance must have a unique name. If no name is assigned when registering a new instance, the instance is given the name "Instance_#" (# is the ID of the instance). If this name already exists, the name "Instance_#. #" is used, in which the second # is a counter that is incremented until the name is unique. The length of the name must be less than DINSTANCE_NAME_LENGTH. See Data types (Page 271).</li> <li>IInstance** out_InstanceInterface: Pointer to a Simulation Runtime interface pointer. The pointer must be initialized with ZERO. The interface is created within the function.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_TIMEOUT	The function does not return on time.
	SREC_WRONG_ARGUMENT	The DLL name, the instance name or the IInstance pointer is invalid.
	SREC_LIMIT_REACHED	There are already 16 instances registered in Runtime Manager.
	SREC_ALREADY_EXISTS	An instance with this name already exists.
Example C++	<pre>ISimulationRuntimeManager * api = ZERO; ERuntimeErrorCode result = Initialize(&amp;api);  // Example: How To Create And Register An Instance IInstance* psa = ZERO; if (result == SREC_OK) {     result = api-&gt;RegisterCustomInstance("C:\\Temp\\vplc.dll"); }</pre>	

**Note**

**Native C++**

If you no longer require the interface, delete it.

See DestroyInterface() (Page 82).

Table 7- 47 RegisterCustomInstance() - .NET (C#)

Syntax	<pre> IInstance RegisterCustomInstance (     string in_VplcDll ); IInstance RegisterCustomInstance (     string in_VplcDll,     string in_InstanceName );                 </pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_VplcDll: The complete path to the DLL of the virtual controller that Siemens.Simatic.Simulation.Runtime.Instance.exe loads at PowerOn.</li> <li>string in_InstanceName: Name to be assigned to the instance. Every instance must have a unique name. If no name is assigned when registering a new instance, the instance is given the name "Instance_#" (# is the ID of the instance). If this name already exists, the name "Instance_#. #" is used, in which the second # is a counter that is incremented until the name is unique. The length of the name must be less than DINSTANCE_NAME_LENGTH. See Data types (Page 271).</li> </ul>	
Return values	If the function is successful, an interface of a virtual controller; otherwise a Null pointer.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.WrongArgument	The name or the ID is invalid.
	ERuntimeErrorCode.LimitReached	There are already 16 instances registered in Runtime Manager.
ERuntimeErrorCode.AlreadyExists	An instance with this name already exists.	

**CreateInterface()**

Creates and returns an interface of an already registered instance of a virtual controller.

The instance could have been registered via the application or another application that uses the Simulation Runtime API.

Table 7- 48 CreateInterface() - Native C++

Syntax	<pre>ERuntimeErrorCode CreateInterface(     WCHAR* in_InstanceName,     IInstance** out_InstanceInterface ); ERuntimeErrorCode CreateInterface(     INT32 in_InstanceID,     IInstance** out_InstanceInterface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• INT32 in_InstanceID: The ID of the registered instance from which you want to receive the interface.</li> <li>• WCHAR* in_InstanceName: The name of the registered instance from which you want to receive the interface.</li> <li>• IInstance** out_InstanceInterface: Pointer to a Simulation Runtime interface pointer. The pointer <b>must</b> be initialized with ZERO. The interface is created within the function.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_TIMEOUT	The function does not return on time.
	SREC_WRONG_ARGUMENT	The name, the ID or the IInstance pointer is invalid.
	SREC_DOES_NOT_EXIST	The instance is not registered in Runtime Manager.
Example C++	<pre>ISimulationRuntimeManager * api = ZERO; ERuntimeErrorCode result = Initialize(&amp;api);  IInstance* psa1 = ZERO; IInstance* psa2 = ZERO; if (result == SREC_OK) {     result = api-&gt;CreateInterface(0, &amp;psa1);     result = api-&gt;CreateInterface(0, &amp;psa2); // psa2 will be the same as psa1 }</pre>	
Example C++	<pre>ISimulationRuntimeManager * api = ZERO; ERuntimeErrorCode result = Initialize(&amp;api);  IInstance* psa = ZERO; if (result == SREC_OK) {     result = api-&gt;CreateInterface(L"My SimulationRuntime Instance",     &amp;psa); }</pre>	

**Note**

**Native C++**

If you no longer require the interface, delete it.

See DestroyInterface() (Page 82).

Table 7- 49 CreateInterface() - .NET (C#)

Syntax	<pre> Instance CreateInterface(     string in_InstanceName ); Instance CreateInterface(     INT32 in_InstanceID );                 </pre>	
Parameters	<ul style="list-style-type: none"> <li>INT32 in_InstanceID: The ID of the registered instance from which you want to receive the interface.</li> <li>string in_InstanceName: The name of the registered instance from which you want to receive the interface.</li> </ul>	
Return values	If the function is successful, an interface of a virtual controller; otherwise a Null pointer.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.WrongArgument	The name or the ID is invalid.
	ERuntimeErrorCode.DoesNotExists	The instance is not registered in Runtime Manager.

### 7.5.3 Remote connections

#### OpenPort()

Opens a port to which another Runtime Manager can connect.

Table 7- 50 OpenPort() - Native C++

Syntax	<code>ERuntimeErrorCode OpenPort(   UINT16 in_Port );</code>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UINT16 in_Port</code>: The port. The value must be greater than 1024.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_TIMEOUT</code>	The function does not return on time.
	<code>SREC_ALREADY_EXISTS</code>	A port is already open.
	<code>SREC_WRONG_ARGUMENT</code>	The port is invalid.
	<code>SREC_CONNECTION_ERROR</code>	The port cannot be opened.

Table 7- 51 OpenPort() - .NET (C#)

Syntax	<code>void OpenPort(   UInt16 in_Port );</code>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UInt16 in_Port</code>: The port. The value must be greater than 1024.</li> </ul>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeErrorCode.AlreadyExists</code>	A port is already open.
	<code>ERuntimeErrorCode.WrongArgument</code>	The port is invalid.
	<code>ERuntimeError-Code.ConnectionError</code>	The port cannot be opened.

### ClosePort()

Closes an open port and all open connections that another Runtime Manager has created to this open port.

Table 7- 52 ClosePort() - Native C++

Syntax	<code>ERuntimeErrorCode ClosePort();</code>	
Parameters	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_TIMEOUT</code>	The function does not return on time.
	<code>SREC_WARNING_INVALID_CALL</code>	No port is open.

Table 7- 53 ClosePort() - .NET (C#)

Syntax	<code>void ClosePort( UInt16 in_Port );</code>	
Parameters	None	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.

### GetPort() / Port { get; }

Returns the open port. If no port is open or the function fails, the return value is 0.

Table 7- 54 GetPort() - Native C++

Syntax	<code>UINT16 GetPort();</code>
Parameters	None
Return values	<code>UINT16</code> : The open port. 0, if no port is open.

Table 7- 55 Port { get; } - .NET (C#)

Syntax	<code>UInt16 Port { get; }</code>
Parameters	None
Return values	<code>UInt16</code> : The open port. 0, if no port is open.
Exceptions	None

**GetRemoteConnectionsCount()**

Supplies the number of open remote connections.

Table 7- 56 GetRemoteConnectionsCount() - Native C++

Syntax	UINT32 GetRemoteConnectionsCount();
Parameters	None
Return values	UINT32: Number of open remote connections.

**GetRemoteConnectionInfoAt()**

Returns information about an open connection.

Table 7- 57 GetRemoteConnectionInfoAt()- Native C++

Syntax	ERuntimeErrorCode GetRemoteConnectionInfoAt (UINT32 in_Index, SConnectionInfo* out_ConnectionInfo);	
Parameters	<ul style="list-style-type: none"> <li>UINT32 in_Index: Index of the connection information that is expected.</li> <li>SConnectionInfo* out_ConnectionInfo: The connection information for this index.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INDEX_OUT_OF_RANGE	Connection information for this index does not exist.

**RemoteConnectionInfo { get; }**

Returns an array of information about all open connections.

Table 7- 58 RemoteConnectionInfo { get; } - .NET (C#)

Syntax	SConnectionInfo[] RemoteConnectionInfo { get; }	
Parameters	None	
Return values	SConnectionInfo[]: An array of information about all open connections.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeErrorCode.Timeout	The function does not return on time.

### RemoteConnect()

Creates a new connection to a remote Runtime Manager or uses an existing connection to create an IRemoteRuntimeManager interface.

Table 7- 59 RemoteConnect() - Native C++

Syntax	<pre>ERuntimeErrorCode RemoteConnect(     UINT8 in_IP3,     UINT8 in_IP2,     UINT8 in_IP1,     UINT8 in_IP0,     UINT16 in_Port,     IRemoteRuntimeManager** out_RemoteRuntimeManagerInterface ERuntimeErrorCode RemoteConnect(     UIP in_IP,     UINT16 in_Port,     IRemoteRuntimeManager** out_RunTimeManagerInterface );</pre>											
Parameters	<ul style="list-style-type: none"> <li>• <b>UINT8 in_IP3:</b> First part of the IP address of the remote PC.</li> <li>• <b>UINT8 in_IP2:</b> Second part of the IP address of the remote PC.</li> <li>• <b>UINT8 in_IP1:</b> Third part of the IP address of the remote PC.</li> <li>• <b>UINT8 in_IP0:</b> Last part of the IP address of the remote PC.</li> <li>• <b>UIP in_IP:</b> IP address of the remote PC.</li> <li>• <b>UINT16 in_Port:</b> The port that is open on the remote PC.</li> <li>• <b>IRemoteRuntimeManager** out_RemoteRuntimeManagerInterface:</b> Pointer to a remote Runtime Manager interface pointer. The pointer <b>must</b> be initialized with <b>ZERO</b>. The interface is created in the function.</li> </ul>											
Return values	<table border="1"> <thead> <tr> <th>Runtime error code</th> <th>Condition</th> </tr> </thead> <tbody> <tr> <td>SREC_OK</td> <td>The function is successful.</td> </tr> <tr> <td>SREC_TIMEOUT</td> <td>The function does not return on time.</td> </tr> <tr> <td>SREC_CONNECTION_ERROR</td> <td>The connection to the remote Runtime Manager cannot be established.</td> </tr> <tr> <td>SREC_WRONG_ARGUMENT</td> <td>IP, port or Instance pointer is invalid.</td> </tr> </tbody> </table>	Runtime error code	Condition	SREC_OK	The function is successful.	SREC_TIMEOUT	The function does not return on time.	SREC_CONNECTION_ERROR	The connection to the remote Runtime Manager cannot be established.	SREC_WRONG_ARGUMENT	IP, port or Instance pointer is invalid.	
Runtime error code	Condition											
SREC_OK	The function is successful.											
SREC_TIMEOUT	The function does not return on time.											
SREC_CONNECTION_ERROR	The connection to the remote Runtime Manager cannot be established.											
SREC_WRONG_ARGUMENT	IP, port or Instance pointer is invalid.											
Example C++	<pre>ISimulationRuntimeManager* api = ZERO; ERuntimeErrorCode result = Initialize(&amp;api);  IRemoteRuntimeManager * client = ZERO;  if (result == SREC_OK) {     result = api-&gt;RemoteConnect(192,203,145,144, 4444, &amp;client); }</pre>											



**Note****Native C++**

If you no longer require the interface, delete it.

See DestroyInterface() (Page 82).

Table 7- 60 RemoteConnect() - .NET (C#)

Syntax	<pre> IRemoteRuntimeManager RemoteConnect (     string in_ConnectionString ); IRemoteRuntimeManager RemoteConnect (     SIP in_IP,     UInt16 in_Port ); IRemoteRuntimeManager RemoteConnect (     Byte in_IP3,     Byte in_IP2,     Byte in_IP1,     Byte in_IP0,     UInt16 In_Port ); </pre>	
Parameters	<ul style="list-style-type: none"> <li>• Byte in_IP3: First part of the IP address of the remote PC.</li> <li>• Byte in_IP2: Second part of the IP address of the remote PC.</li> <li>• Byte in_IP1: Third part of the IP address of the remote PC.</li> <li>• Byte in_IP0: Last part of the IP address of the remote PC.</li> <li>• string in_ConnectionString: A string in the form of "&lt;IP3&gt;.&lt;IP2&gt;.&lt;IP1&gt;.&lt;IP0&gt;:&lt;Port&gt;" Example: "182.203.145.144:4444".</li> <li>• SIP in_IP: IP address of the remote PC.</li> <li>• UInt16 in_Port: The port that is open on the remote PC.</li> </ul>	
Return values	IRemoteRuntimeManager: Interface to the remote Runtime Manager.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.ConnectionError	Connection to the remote Runtime Manager cannot be established.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.WrongArgument	IP or port is invalid.

**See also**

Data types (Page 257)

Tag list (Page 135)

**7.5.4 Events**

**Events for the ISimulationRuntimeManager interface**

The following events are differentiated in the API:

Table 7- 61 Events for the ISimulationRuntimeManager interface

Event	Cause
OnConfigurationChanged (Page 106)	This event is triggered at a change of the Runtime Manager configuration. For example, when a new instance is registered, an instance is removed or a connection to a client is established.  The Control Panel uses such an event to update the list of available instances.
OnRuntimeManagerLost (Page 110)	The event is triggered when the connection to Runtime Manager is interrupted.

**7.5.4.1 OnConfigurationChanged**

**OnConfigurationChanged**

Registers or unregisters an event handler method.

Table 7- 62 OnConfigurationChanged - .NET (C#)

Syntax	<code>event Delegate_SRCC_UINT32_UINT32_INT32 OnConfigurationChanged;</code>
Parameters	None. See Delegate_SRCC_UINT32_UINT32_INT32 (Page 269).
Return values	None
Exceptions	None
Note	The event handler method runs in a separate thread.

**RegisterOnConfigurationChangedCallback()**

When the event occurs, the registered callback function is called. Only one callback function can be registered for the event. The registration of a new callback function causes the previous callback function to be deleted.

Table 7- 63 RegisterOnConfigurationChangedCallback() - Native C++

Syntax	<code>void RegisterOnConfigurationChangedCallback( _EventCallback_SRCC_UINT32_UINT32_INT32 in_CallbackFunction );</code>
Parameters	<ul style="list-style-type: none"> <li>• <code>_EventCallback_SRCC_UINT32_UINT32_INT32 in_CallbackFunction:</code> A callback function to subscribe to an event. See <code>_EventCallback_SRCC_UINT32_UINT32_INT32</code> (Page 263).</li> </ul>
Return values	None
Note	The event handler method runs in a separate thread.

**RegisterOnConfigurationChangedEvent()**

When the event occurs, the registered event object is set to the signaled state. Only one event object can be registered for the event. Registration of a new event object causes the previous event object to be deleted.

Table 7- 64 RegisterOnConfigurationChangedEvent() - Native C++

Syntax	<code>void RegisterOnConfigurationChangedEvent(); void RegisterOnConfigurationChangedEvent( HANDLE* in_Event );</code>
Parameters	<ul style="list-style-type: none"> <li>• None: An internal event object is registered.</li> <li>• <code>HANDLE* in_Event:</code> A handle for a user-specific event object. The event object is registered.</li> </ul>
Return values	None

Table 7- 65 RegisterOnConfigurationChangedEvent() - .NET (C#)

Syntax	<code>void RegisterOnConfigurationChangedEvent();</code>
Parameters	None
Return values	None

### UnregisterOnConfigurationChangedCallback()

Unregisters the callback function. When the event occurs, no callback function is called.

Table 7- 66 UnregisterOnConfigurationChangedCallback() - Native C++

Syntax	<code>void UnregisterOnConfigurationChangedCallback();</code>
Parameters	None
Return values	None

### UnregisterOnConfigurationChangedEvent()

Unregisters the event object.

Table 7- 67 UnregisterOnConfigurationChangedEvent() - Native C++

Syntax	<code>void UnregisterOnConfigurationChangedEvent();</code>
Parameters	None
Return values	None

Table 7- 68 UnregisterOnConfigurationChangedEvent() - .NET (C#)

Syntax	<code>void UnregisterOnConfigurationChangedEvent();</code>
Parameters	None
Return values	None

**WaitForOnConfigurationChangedEvent()**

The function blocks the program until the registered event object is set to the signaled state or the timeout interval is exceeded.

Table 7- 69 WaitForOnConfigurationChangedEvent() - Native C++

Syntax	<pre>bool WaitForOnConfigurationChangedEvent(); bool WaitForOnConfigurationChangedEvent(     UINT32 in_Time_ms );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UINT32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

Table 7- 70 WaitForOnConfigurationChangedEvent - .NET (C#)

Syntax	<pre>bool WaitForOnConfigurationChangedEvent(); bool WaitForOnConfigurationChangedEvent(     UInt32 in_Time_ms );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UInt32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

### 7.5.4.2 OnRuntimeManagerLost

#### OnRuntimeManagerLost

Registers or unregisters an event handler method.

Table 7- 71 OnRuntimeManagerLost - .NET (C#)

Syntax	<code>event Delegate_Void OnRuntimeManagerLost;</code>
Parameters	None. See Delegate_Void (Page 265).
Return values	None
Exceptions	None
Note	The event handler method runs in a separate thread.

#### RegisterOnRuntimeManagerLostCallback()

When the event occurs, the registered callback function is called. Only one callback function can be registered for the event. The registration of a new callback function causes the previous callback function to be deleted.

Table 7- 72 RegisterOnRuntimeManagerLostCallback() - Native C++

Syntax	<code>void RegisterOnRuntimeManagerLostCallback( EventCallback_VOID in_CallbackFunction );</code>
Parameters	<ul style="list-style-type: none"> <li>EventCallback_VOID in_CallbackFunction: A callback function that subscribes to the event. See EventCallback_VOID (Page 259).</li> </ul>
Return values	None
Note	The event handler method runs in a separate thread.

**RegisterOnRuntimeManagerLostEvent()**

When the event occurs, the registered event object is set to the signaled state. Only one event object can be registered for the event. Registration of a new event object causes the previous event object to be deleted.

Table 7- 73 RegisterOnRuntimeManagerLostEvent() - Native C++

Syntax	<pre>void RegisterOnRuntimeManagerLostEvent(); void RegisterOnRuntimeManagerLostEvent( HANDLE* in_Event );</pre>
Parameters	<ul style="list-style-type: none"> <li>• None: An internal event handle is registered.</li> <li>• HANDLE* in_Event: A user-specific event handle is registered.</li> </ul>
Return values	None

Table 7- 74 RegisterOnRuntimeManagerLostEvent() - .NET (C#)

Syntax	<pre>void RegisterOnRuntimeManagerLostEvent();</pre>
Parameters	None
Return values	None

**UnregisterOnRuntimeManagerLostCallback()**

Unregisters the callback function. When the event occurs, no callback function is called.

Table 7- 75 UnregisterOnRuntimeManagerLostCallback() - Native C++

Syntax	<pre>void UnregisterOnRuntimeManagerLostCallback();</pre>
Parameters	None
Return values	None

### UnregisterOnRuntimeManagerLostEvent()

Unregisters the event object.

Table 7- 76 UnregisterOnRuntimeManagerLostEvent() - Native C++

Syntax	<code>void UnregisterOnRuntimeManagerLostEvent();</code>
Parameters	None
Return values	None

Table 7- 77 UnregisterOnRuntimeManagerLostEvent() - .NET (C#)

Syntax	<code>void UnregisterOnRuntimeManagerLostEvent();</code>
Parameters	None
Return values	None

### WaitForOnRuntimeManagerLostEvent()

The function will block the program until the registered event object is set to the signaled state or the timeout interval is exceeded.

Table 7- 78 WaitForOnRuntimeManagerLostEvent() - Native C++

Syntax	<code>bool WaitForOnRuntimeManagerLostEvent(); bool WaitForOnRuntimeManagerLostEvent(   UINT32 in_Time_ms );</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UINT32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined timeout interval.</li> </ul>

Table 7- 79 WaitForOnRuntimeManagerLostEvent() - .NET (C#)

Syntax	<code>bool WaitForOnRuntimeManagerLostEvent(); bool WaitForOnRuntimeManagerLostEvent(   UInt32 in_Time_ms );</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UInt32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined timeout interval.</li> </ul>



## 7.6 API Instances

### 7.6.1 Interfaces - Information and settings

#### Dispose()

Deletes the managed interface and unloads the native components of the user interfaces.

Table 7- 80 Dispose() - .NET (C#)

Syntax	<code>void Dispose()</code>
Parameters	None
Return values	None

#### GetID() / ID { get; }

Returns the instance ID. The ID is assigned by Runtime Manager when the instance is registered.

Table 7- 81 GetID() - Native C++

Syntax	<code>INT32 GetID();</code>
Parameters	None
Return values	<code>INT32: Instance ID</code>

Table 7- 82 ID { get; } - .NET (C#)

Syntax	<code>UInt32 ID { get; }</code>
Parameters	None
Return values	<code>UInt32: Instance ID</code>
Exceptions	None

**GetName() / Name { get; }**

Returns the name of the instance.

Table 7- 83 GetName() - Native C++

Syntax	<pre>ERuntimeErrorCode GetName (     WCHAR inout_Name[],     UINT32 in_ArrayLength );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR inout_Name[]:                     <p>A user-allocated storage for the name of the instance. The field length should be at least as long as <code>DINSTANCE_NAME_MAX_LENGTH</code>. See Definitions and constants (Page 271).</p> </li> <li>UINT32 in_ArrayLength:                     <p>Field length (Wide character)</p> </li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_WRONG_ARGUMENT	The name does not fit in the storage.
Example C++	<pre>ISimulationRuntimeManager * api = NULL; ERuntimeErrorCode result = Initialize(&amp;api);  IInstance* psa = NULL; if (result == SREC_OK) {     result = api-&gt;RegisterInstance(&amp;psa); }  WCHAR name[DINSTANCE_NAME_MAX_LENGTH]; if (result == SREC_OK) {     result = psa-&gt;GetName(name, DINSTANCE_NAME_MAX_LENGTH); } }</pre>	

Table 7- 84 Name { get; } - .NET (C#)

Syntax	string Name { get; }	
Parameters	None	
Return values	Name of the instance.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.

## GetCPUType()

Returns the CPU type of the virtual controller.

Table 7- 85 GetCPUType() - Native C++

Syntax	<code>ECPUType GetCPUType();</code>
Parameters	None
Return values	An enumeration element that defines the CPU type. See ECPUType (Page 289).

## SetCPUType()

Sets the CPU type of the virtual controller. A change of CPU type occurs only when the controller is restarted.

Table 7- 86 SetCPUType() - Native C++

Syntax	<code>void SetCPUType(ECPUType in_Value);</code>
Parameters	<ul style="list-style-type: none"> <li><code>ECPUType in_Value:</code> Defines which CPU type is simulated at the start of the instance.  When a different CPU type is loaded via STEP 7 or from the Virtual Memory Card, this CPU type applies.</li> </ul>
Return values	None

## CPUType { get; set; }

Returns or sets the CPU type of the virtual controller. A change of CPU type occurs only when the controller is restarted.

When a different CPU type is loaded via STEP 7 or from the Virtual Memory Card, this CPU type applies.

Table 7- 87 CPUType { get; set; } - .NET (C#)

Syntax	<code>ECPUType CPUType { get; set; }</code>
Parameters	None
Return values	An enumeration element that defines the CPU type.
Exceptions	None

### GetCommunicationInterface()

Returns the communication interface of the virtual controller: Local communication (Softbus) or TCPIP. A change of communication interface occurs only when the controller is restarted. All instances that are started must use the same communication interface.

PowerOn is prevented if a communication interface that is not used by the started instances is selected.

Table 7- 88 GetCommunicationInterface() - Native C++

Syntax	<code>ECommunicationInterface GetCommunicationInterface();</code>
Parameters	None
Return values	<ul style="list-style-type: none"> <li>• SRCI_NONE Cannot be selected. Is returned if the instance interface is no longer valid.</li> <li>• SRCI_SOFTBUS Is returned if the virtual controller uses the Softbus.</li> <li>• SRCI_TCPIP Is returned if the virtual controller communicates over the virtual adapter.</li> </ul>

### SetCommunicationInterface()

Sets the communication interface of the virtual controller: Local communication (Softbus) or TCPIP. A change of communication interface occurs only when the controller is restarted. All instances that are started must use the same communication interface.

PowerOn is prevented if a communication interface that is not used by the started instances is selected.

Table 7- 89 SetCommunicationInterface() - Native C++

Syntax	<code>void SetCommunicationInterface(ECommunicationInterface in_Value);</code>
Parameters	<ul style="list-style-type: none"> <li>• SRCI_NONE Cannot be selected.</li> <li>• SRCI_SOFTBUS Is set to activate communication via Softbus.</li> <li>• SRCI_TCPIP Is set to activate communication over the virtual adapter.</li> </ul>
Return values	None

**CommunicationInterface { get; set; }**

Sets or returns the communication interface of the virtual controller: Local communication (Softbus) or TCPIP. A change of communication interface occurs only when the controller is restarted. All instances that are started must use the same communication interface.

PowerOn is prevented if a communication interface that is not used by the started instances is selected.

Table 7- 90 CommunicationInterface { get; set; } - .NET (C#)

Syntax	ECommunicationInterface CommunicationInterface { get; set; }
Parameters	None
Return values	<ul style="list-style-type: none"> <li>• ECommunicationInterface.None Cannot be selected. Is returned if the instance interface is no longer valid.</li> <li>• ECommunicationInterface.Softbus Is returned if the virtual controller uses the Softbus.</li> <li>• ECommunicationInterface.TCPIP Is returned if the virtual controller communicates over the virtual adapter.</li> </ul>
Exceptions	None

**GetInfo() / Info { get; }**

Returns a structure that provides information about the instance.

Table 7- 91 GetInfo() - Native C++

Syntax	SInstanceInfo GetInfo();
Parameters	None
Return values	SInstanceInfo: A structure that provides information about the instance. See SInstanceInfo (Page 278).

Table 7- 92 Info { get; } - .NET (C#)

Syntax	SInstanceInfo Info { get; }
Parameters	None
Return values	SInstanceInfo: A structure that provides information about the instance.
Exceptions	None

### UnregisterInstance()

Unregisters this instance from Runtime Manager.

**Note**

**Loss of the interfaces**

Other applications that are connected to this instance will lose their interface to this instance.

Table 7- 93 UnregisterInstance() - Native C++

Syntax	<code>ERuntimeErrorCode UnregisterInstance();</code>	
Parameters	None	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.

Table 7- 94 UnregisterInstance() - .NET (C#)

Syntax	<code>void UnregisterInstance();</code>	
Parameters	None	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.

## 7.6.2 Controller - Information and settings

### GetControllerName() / ControllerName { get; }

Returns the downloaded name of the virtual controller.

Table 7- 95 GetControllerName() - Native C++

Syntax	<pre>ERuntimeErrorCode GetControllerName(     WCHAR inout_Name[],     UINT32 in_ArrayLength );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR inout_Name[]: A user-allocated storage for the name.</li> <li>UINT32 in_ArrayLength: The length of the storage.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INDEX_OUT_OF_RANGE	The name does not fit in the storage.

Table 7- 96 ControllerName { get; } - .NET (C#)

Syntax	string ControllerName { get; }	
Parameters	None	
Return values	string: The downloaded name of the virtual controller.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.

**GetControllerShortDesignation() / ControllerShortDesignation { get; }**

Returns the downloaded short designation of the virtual controller.

Table 7- 97 GetControllerShortDesignation() - Native C++

Syntax	ERuntimeErrorCode GetControllerShortDesignation( WCHAR inout_ShortDesignation[], UINT32 in_ArrayLength );	
Parameters	<ul style="list-style-type: none"> <li>WCHAR inout_ShortDesignation[]: A user-allocated storage for the short designation.</li> <li>UINT32 in_ArrayLength: The length of the storage.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INDEX_OUT_OF_RANGE	The name does not fit in the storage.

Table 7- 98 ControllerShortDesignation { get; } - .NET (C#)

Syntax	string ControllerShortDesignation { get; }	
Parameters	None	
Return values	string: The downloaded short designation of the virtual controller.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.

**GetControllerIPCount()**

Returns the number of configured IP addresses of the virtual controller. If the function fails, the return value is 0.

Table 7- 99 GetControllerIPCount() - Native C++

Syntax	UINT32 GetControllerIPCount();
Parameters	None
Return values	INT32: Number of configured IP addresses of the virtual controller.



**GetControllerIP() / ControllerIP { get; }**

Returns a configured IP address of the instance.

Table 7- 100 GetControllerIP() - Native C++

Syntax	<pre>UIP GetControllerIP(); UIP GetControllerIP(     UINT32 in_Index );</pre>
Parameters	<ul style="list-style-type: none"> <li>WCHAR in_Index:</li> </ul> <p>The index of the IP address you want to receive. The index must be less than the value you receive from <code>GetControllerIPCount()</code>. The default setting is 0.</p>
Return values	UIP: IP address of the virtual controller. If the function fails, the return value is 0.

Table 7- 101 ControllerIP { get; } - .NET (C#)

Syntax	<code>string[] ControllerIP { get; }</code>
Parameters	None
Return values	<code>string</code> : All downloaded IP addresses of the virtual controller. If the function fails, the field is empty.
Exceptions	None

**GetControllerIPSuite4() / ControllerIPSuite4 { get; }**

Returns the IP suite instance. If the "Softbus" communication interface is used, the subnet mask and default gateway are 0.

Table 7- 102 GetControllerIPSuite4() Native C++

Syntax	<pre>SIPSuite4 GetControllerIPSuite4(); SIPSuite4 GetControllerIPSuite4(     UINT32 in_Index );</pre>
Parameters	<ul style="list-style-type: none"> <li>WCHAR in_Index:</li> </ul> <p>The index of the IP address you want to receive. The index must be less than the value you receive from <code>GetControllerIPCount()</code>. The default setting is 0.</p>
Return values	SIPSuite4: The IP suite of the virtual controller. If the function fails, the return values are 0.

Table 7- 103 ControllerIPSuite4 { get; } - .NET (#)

Syntax	<code>SIPSuite4[] ControllerIPSuite4 { get; };</code>
Parameters	None
Return values	<code>SIPSuite4[]</code> : All downloaded IP suites of the virtual controller. If the function fails, the field is empty.
Exceptions	None

**SetIPSuite()**

Sets the IP suite of the network interface of a virtual controller.

Table 7- 104 SetIPSuite() - Native C++

Syntax	<pre>ERuntimeErrorCode SetIPSuite(     UINT32 in_InterfaceID,     SIPSuite4 in_IPSuite,     bool in_IsRemanent );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <b>UINT32 in_InterfaceID:</b> The ID of the network interface.</li> <li>• <b>SIPSuite4 in_IPSuite:</b> The IP suite that is to be assigned to the network interface. The IP suite contains the IP address, the subnet mask and the standard gateway. If the communication interface is "Softbus", the subnet mask and standard gateway are ignored.</li> <li>• <b>bool in_IsRemanent:</b> If <code>true</code>, the IP suite is saved after restart of the virtual controller. If the communication interface is "Softbus", this flag is ignored.</li> </ul>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	There is no network interface with this ID.
	SREC_INVALID_OPERATING_STATE	The virtual controller has not yet completed the boot process or is already in the shutdown phase.

Table 7- 105 SetIPSuite() - .NET (C#)

Syntax	<pre>void SetIPSuite(     UInt32 in_InterfaceID,     SIPSuite4 in_IPSuite,     bool in_IsRemanent );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <b>UInt32 in_InterfaceID:</b> The ID of the network interface.</li> <li>• <b>SIPSuite4 in_IPSuite:</b> If the communication interface is "Softbus", the subnet mask and standard gateway are ignored.</li> <li>• <b>bool in_IsRemanent:</b> If <code>true</code>, the IP suite is saved after restart of the virtual controller. If the communication interface is "Softbus", this flag is ignored.</li> </ul>	

Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeErrorCode.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	There is no network interface with this ID.
	<code>ERuntimeErrorCode.InvalidOperatingState</code>	The virtual controller has not yet completed the boot process or is already in the shutdown phase.

## GetStoragePath()

Returns the full directory in which the instance stores its data.

Table 7- 106 GetStoragePath() - Native C++

Syntax	<code>ERuntimeErrorCode GetStoragePath( WCHAR inout StoragePath[], UINT32 in_ArrayLength );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>WCHAR inout_StoragePath[]</code>: A user-allocated storage for the storage path. The length of the array should be at least as long as <code>DSTORAGE_PATH_MAX_LENGTH</code>. See Data types (Page 257).</li> <li><code>UINT32 in_ArrayLength</code>: Length of the array (Wide character)</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The function does not return on time.
	<code>SREC_INDEX_OUT_OF_RANGE</code>	The path does not fit in the storage.

### SetStoragePath()

Sets the full path of the directory in which the instance stores its data. This can also be a network share.

Set the path before you start the instance. A change to the path takes effect only when the controller is restarted.

If no path is set, the default setting:

<My Documents>>\Siemens\Simatic\Simulation\Runtime\Persistence\<Instance Name> is used.

Table 7- 107 SetStoragePath() - Native C++

Syntax	ERuntimeErrorCode SetStoragePath( WCHAR* in_StoragePath );	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_StoragePath:  Full name of the storage path. The length of the name must be less than <code>DSTORAGE_PATH_MAX_LENGTH</code>. See <a href="#">Data types (Page 257)</a>.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INDEX_OUT_OF_RANGE	The length of the path exceeds the limit.
	SREC_WRONG_ARGUMENT	The path contains invalid characters.

### StoragePath { get; set; }

Returns or sets the full path of the directory in which the instance stores its retentive data. This can also be a network share.

Set the path before you start the instance. A change to the path takes effect only when the controller is restarted.

If no path is set, the default setting:

<My Documents>>\Siemens\Simatic\Simulation\Runtime\Persistence\<Instance Name> is used.

Table 7- 108 StoragePath { get; set; } - .NET (C#)

Syntax	string StoragePath { get; set; }	
Parameters	None	
Return values	string: The configured storage path.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.IndexOutOfRange	The length of the path exceeds the limit.
	ERuntimeErrorCode.WrongArgument	The path contains invalid characters.

## ArchiveStorage()

The user program, the hardware configuration and the retentive data are stored in a file, the Virtual SIMATIC Memory Card. `ArchiveStorage()` stores this file as a ZIP file. The instance of the virtual controller must be in OFF operating state for this.

Table 7- 109 ArchiveStorage() - Native C++

Syntax	<code>ERuntimeErrorCode ArchiveStorage( WCHAR* in_FullFileName );</code>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR in_FullFileName: The full path to the ZIP file.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INVALID_OPERATING_STATE	The instance is not in OFF operating state.
	SREC_INVALID_ARCHIVE_PATH	The archive path is invalid.
	SREC_CREATE_DIRECTORIES_FAILED	The directory for the ZIP file could not be created.
	SREC_ARCHIVE_STORAGE_FAILED	The ZIP file could not be created.

Table 7- 110 ArchiveStorage() - .NET (C#)

Syntax	<code>void ArchiveStorage( string in_ArchiveStorageFile );</code>	
Parameters	<ul style="list-style-type: none"> <li>string in_ArchiveStorageFile: The full path to the ZIP file. The path relates to directories of the computer on which the instance runs.</li> </ul>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeError-Code.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InvalidOperatingState</code>	The instance is not in OFF operating state.
	<code>ERuntimeError-Code.InvalidArchivePath</code>	The archive path is invalid.
	<code>ERuntimeError-Code.CreateDirectoriesFailed</code>	The directory for the ZIP file could not be created.
	<code>ERuntimeError-Code.ArchiveStorageNotCreated</code>	The ZIP file could not be created.

### RetrieveStorage()

RetrieveStorage() creates a Virtual SIMATIC Memory Card from the archived ZIP file. The virtual controller must be in OFF operating state for this.

Table 7- 111 RetrieveStorage() - Native C++

Syntax	<code>ERuntimeErrorCode RetrieveStorage( WCHAR* in_FullFileName );</code>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_FullFileName: The full path to the ZIP file. The path relates to directories of the computer on which the instance runs.</li> </ul>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INVALID_OPERATING_STATE	The instance is not in OFF operating state.
	SREC_INVALID_ARCHIVE_PATH	The archive path is invalid.
	SREC_DELETE_EXISTING_STORAGE_FAILED	The old storage cannot be deleted.
	SREC_RETRIEVE_STORAGE_FAILURE	The ZIP file cannot be unzipped.

Table 7- 112 RetrieveStorage() - .NET (C#)

Syntax	<code>void RetrieveStorage( string in_ArchiveStorageFile );</code>	
Parameters	<ul style="list-style-type: none"> <li>string in_ArchiveStorageFile: The full path to the ZIP file.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InvalidOperatingState	The instance is not in OFF operating state.
	ERuntimeError-Code.InvalidArchivePath	The archive path is invalid.
	ERuntimeError-Code.DeleteExistingStorageFailed	The old storage cannot be deleted.
	ERuntimeError-Code.RetrieveStorageFailure	The ZIP file cannot be unzipped.

### 7.6.3 Operating state

#### GetOperatingState() / OperatingState { get; }

Returns the operating state of the virtual controller. If the operating state changes, the event `OnOperatingStateChanged()` (Page 228) is triggered. For details about the operating state, see Data types (Page 287).

Table 7- 113 GetOperatingState() - Native C++

Syntax	<code>EOperatingState GetOperatingState();</code>
Parameters	None
Return values	<ul style="list-style-type: none"> <li>• <code>SROS_INVALID_OPERATING_STATE:</code> If the function fails.</li> <li>• <code>SROS_OFF:</code> If the Simulation Runtime instance is not running.</li> <li>• <code>SROS_BOOTING:</code> If <code>PowerOn()</code> was called while in this state and the virtual controller is not yet ready to start the user program.</li> <li>• <code>SROS_STOP:</code> If the virtual controller is in STOP state.</li> <li>• <code>SROS_STARTUP:</code> If the user program is currently changing from STOP to RUN.</li> <li>• <code>SROS_RUN:</code> If the user program is running.</li> <li>• <code>SROS_FREEZE:</code>  If the user program is being stopped (Freeze status).</li> <li>• <code>SROS_SHUTTING_DOWN:</code> If <code>PowerOff()</code> was called but the virtual controller is still in the Shutdown phase.</li> </ul>

Table 7- 114 OperatingState { get; } - .NET (C#)

Syntax	EOperatingState OperatingState { get; }
Parameters	None
Return values	<ul style="list-style-type: none"> <li>• EOperatingState.InvalidOperatingState: If the function fails.</li> <li>• EOperatingState.Off: If the Simulation Runtime instance is not running.</li> <li>• EOperatingState.Booting: If PowerOn() was called while in this state and the virtual controller is not yet ready to start the user program.</li> <li>• EOperatingState.Stop: If the virtual controller is in STOP state.</li> <li>• EOperatingState.Startup: If the user program is currently changing from STOP to RUN.</li> <li>• EOperatingState.Run: If the user program is running.</li> <li>• EOperatingState.Freeze: If the user program is being stopped (Freeze status).</li> <li>• EOperatingState.ShuttingDown: If PowerOff() was called but the virtual controller is still in the Shutdown phase.</li> </ul>



## PowerOn()

The function creates the process for the Simulation Runtime instance and starts the firmware of the virtual controller.

Table 7- 115 PowerOn() - Native C++

Syntax	<pre>ERuntimeErrorCode PowerOn(); ERuntimeErrorCode PowerOn(     UINT32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UINT32 in_Timeout_ms</code>: A timeout value in milliseconds. <ul style="list-style-type: none"> <li>– If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>– If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> <li>– Expected operating states if this function is successful: { <code>SROS_STOP</code> , <code>SROS_RUN</code> }</li> </ul> </li> </ul>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The expected operating state does not occur on time.
	<code>SREC_ERROR_LOADING_DLL</code>	The Siemens.Simatic.Simulation.Runtime.Instance.exe cannot load the Siemens.Simatic.PlcSim.Vplc1500.dll.
	<code>SREC_STORAGE_PATH_ALREADY_IN_USE</code>	The selected path for this instance is already being used by another instance.
	<code>SREC_NO_STORAGE_PATH_SET</code>	The path could not be created. The length of the <code>DSTORAGE_PATH_MAX_LENGTH</code> characters may be exceeded.
	<code>SREC_WARNING_ALREADY_EXISTS</code>	Only one warning. The instance is started.
	<code>SREC_WARNING_TRIAL_MODE_ACTIVE</code>	No license available. You can use the instance without restrictions for a period of one hour. Afterwards, the instance is shut down.  The warning occurs even when you import a license within this hour. The instance is then not shut down.
	<code>SREC_VIRTUAL_SWITCH_MISCONFIGURED</code>	The virtual switch is configured incorrectly.
<code>SREC_INSTANCE_NOT_RUNNING</code>	The process of the virtual controller is no longer running.	

Table 7- 116 PowerOn() - .NET (C#)

Syntax	<pre>ERuntimeErrorCode PowerOn(); ERuntimeErrorCode PowerOn(     UInt32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• UInt32 in_Timeout_ms:                     <p>A timeout value in milliseconds.</p> <ul style="list-style-type: none"> <li>– If no timeout value is set, the function returns immediately. Subscribe to the OnOperatingStateChanged() event to find out when the operation has been completed.</li> <li>– If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> <p>Expected operating states when this function is successful:</p> <pre>{ EOperatingState.Run, EOperatingState.Stop }</pre> </li> </ul>	
Return values	Runtime error code	Condition
	ERuntimeErrorCode.OK	The function is successful.
	ERuntimeError-Code.WarningAlreadyExists	Only one warning. The instance is started.
	ERuntimeError-Code.WarningTrialModeActive	<p>No license available. You can use the instance without restrictions for a period of one hour. Afterwards, the instance is shut down.</p> <p>The warning occurs even when you import a license within this hour. The instance is then not shut down.</p>
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The expected operating state does not occur on time.
	ERuntimeError-Code.ErrorLoadingDll	The Siemens.Simatic.Simulation.Runtime.Instance.exe cannot load the Siemens.Simatic.PlcSim.Vplc1500.dll.
	ERuntimeError-Code.StoragePathAlreadyInUse	The selected path for this instance is already being used by another instance.
	ERuntimeError-Code.NoStoragePathSet	The path could not be created. The length of the <code>DSTORAGE_PATH_MAX_LENGTH</code> characters may be exceeded.
	ERuntimeError-Code.VirtualSwitchMisconfigured	The virtual switch is configured incorrectly.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is no longer running.

**PowerOff()**

Shuts down the Simulation Runtime and closes its process.

Table 7- 117 PowerOff() - Native C++

Syntax	<pre>ERuntimeErrorCode PowerOff(); ERuntimeErrorCode PowerOff(     UINT32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UINT32 in_Timeout_ms</code>: A timeout value in milliseconds. <ul style="list-style-type: none"> <li>– If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>– If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> </li> </ul> <p>Expected operating states when this function is successful: { <code>SROS_OFF</code> }</p>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The expected operating state does not occur on time.
	<code>SREC_INSTANCE_NOT_RUNNING</code>	The process of the virtual controller is not running.

Table 7- 118 PowerOff() - .NET (C#)

Syntax	<pre>void PowerOff(); void PowerOff(     UInt32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UInt32 in_Timeout_ms</code>: A timeout value in milliseconds. <ul style="list-style-type: none"> <li>– If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>– If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> </li> </ul> <p>Expected operating states when this function is successful: { <code>EOperatingState.Off</code> }</p>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeErrorCode.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The expected operating state does not occur on time.
	<code>ERuntimeErrorCode.InstanceNotRunning</code>	The process of the virtual controller is not running.

### MemoryReset()

Shuts down the virtual controller, closes its processes and performs a restart.

Table 7- 119 MemoryReset() - Native C++

Syntax	<pre>ERuntimeErrorCode MemoryReset(); ERuntimeErrorCode MemoryReset(     UINT32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UINT32 in_Timeout_ms</code>: A timeout value in milliseconds.                     <ul style="list-style-type: none"> <li>- If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>- If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> </li> </ul> <p>Expected operating states when this function is successful: { <code>SROS_STOP</code> , <code>SROS_RUN</code> }</p>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The expected operating state does not occur on time.
	<code>SREC_INSTANCE_NOT_RUNNING</code>	The process of the virtual controller is not running.

Table 7- 120 MemoryReset() - .NET (C#)

Syntax	<pre>void MemoryReset(); void MemoryReset(     UInt32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UInt32 in_Timeout_ms</code>: A timeout value in milliseconds.                     <ul style="list-style-type: none"> <li>- If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>- If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> </li> </ul> <p>Expected operating states when this function is successful: { <code>EOperatingState.Run</code> , <code>EOperatingState.Stop</code> }</p>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeErrorCode.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The expected operating state does not occur on time.
	<code>ERuntimeErrorCode.InstanceNotRunning</code>	The process of the virtual controller is not running.

**Run()**

Calls on the virtual controller to change to RUN operating state.

Table 7- 121 Run() - Native C++

Syntax	<pre>ERuntimeErrorCode Run(); ERuntimeErrorCode Run(     UINT32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UINT32 in_Timeout_ms</code>: A timeout value in milliseconds. <ul style="list-style-type: none"> <li>– If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>– If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> </li> </ul> <p>Expected operating states when this function is successful: { <code>SROS_STOP</code> , <code>SROS_RUN</code> }</p>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The expected operating state does not occur on time.
	<code>SREC_INSTANCE_NOT_RUNNING</code>	The process of the virtual controller is not running.

Table 7- 122 Run() - .NET (C#)

Syntax	<pre>void Run(); void Run(     UInt32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UInt32 in_Timeout_ms</code>: A timeout value in milliseconds. <ul style="list-style-type: none"> <li>– If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>– If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> </li> </ul> <p>Expected operating states when this function is successful: { <code>EOperatingState.Run</code> }</p>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeError-Code.Timeout</code>	The expected operating state does not occur on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.

Stop()

Calls on the virtual controller to change to STOP operating state.

Table 7- 123 Stop() - Native C++

Syntax	<pre>ERuntimeErrorCode Stop(); ERuntimeErrorCode Stop(     UINT32 in_Timeout_ms );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UINT32 in_Timeout_ms</code>: A timeout value in milliseconds.                     <ul style="list-style-type: none"> <li>– If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>– If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> </li> </ul> <p>Expected operating states when this function is successful: { <code>SROS_STOP</code> }</p>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The expected operating state does not occur on time.
	<code>SREC_INSTANCE_NOT_RUNNING</code>	The process of the virtual controller is not running.

Table 7- 124 Stop() - .NET (C#)

Syntax	<pre>void Stop(); void Stop(     bool in_IsSynchronous );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UInt32 in_Timeout_ms</code>: A timeout value in milliseconds.                     <ul style="list-style-type: none"> <li>– If no timeout value is set, the function returns immediately. Subscribe to the <code>OnOperatingStateChanged()</code> event to find out when the operation has been completed.</li> <li>– If the value is greater than 0 (a value of 60000 is recommended), the function returns when the operation has been completed or after a timeout.</li> </ul> </li> </ul> <p>Expected operating states when this function is successful: { <code>EOperatingState.Stop</code> }</p>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The expected operating state does not occur on time.
	<code>ERuntimeErrorCode.InstanceNotRunning</code>	The process of the virtual controller is not running.

## 7.6.4 Tag list

### UpdateTagList()

The function reads the tags from the virtual controller and writes them to the shared storage arranged by name.

If the tag is an array or a structure, there are multiple entries.

In the case of a structure, there is an entry for the structure itself and an additional entry for each structure element.

```
Entry_1: "StructName"  
Entry_2: "StructName.ElementName_1"  
..  
Entry_N: "StructName.ElementName_n"
```

In the case of an array, in this example a two-dimensional array, there is an entry for the array itself and an additional entry for each array element.

```
Entry_1: "ArrayName"  
Entry_2: "ArrayName[a,b]", where {a} and {b} correspond to the first index of the  
respective dimension  
..  
Entry_N: "ArrayName[x,y]", where {x} and {y} correspond to the last index of the  
respective dimension
```

Memory for up to 500000 entries (not PLC tags) is reserved for the list. If the list becomes too large, the function returns the error/exception "NOT\_ENOUGH\_MEMORY".

If there are problems with the maximum number of entries and not all tags are needed, two filters can be used when refreshing the tag table.

Table 7- 125 UpdateTagList() - Native C++

Syntax	<pre>ERuntimeErrorCode UpdateTagList(); ERuntimeErrorCode UpdateTagList(     ETagListDetails in_TagListDetails ); ERuntimeErrorCode UpdateTagList(     ETagListDetails in_TagListDetails,     bool in_IsHMIVisibleOnly );</pre>	
Parameters	<ul style="list-style-type: none"> <li>ETagListDetails in_TagListDetails: Every combination of the following four areas: <b>IO:</b> Inputs and Outputs <b>M:</b> Bit memory <b>CT:</b> Counters and Timers <b>DB:</b> Data Blocks  <b>Example: IOM</b> reads only the tags from the area Inputs / Outputs and Bit memory. The default setting is <b>IOMCTDB</b>.</li> <li>bool in_IsHMIVisibleOnly: If true, only tags marked with "HMI Visible" are read. The default setting is true.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
SREC_NOT_ENOUGH_MEMORY	More than 50000 entries are requested.	



Table 7- 126 UpdateTagList() - .NET (C#)

Syntax	<pre>void UpdateTagList(); void UpdateTagList(     ETagListDetails in_TagListDetails ); void UpdateTagList(     ETagListDetails in_TagListDetails,     bool in_IsHMIVisibleOnly );</pre>	
Parameters	<ul style="list-style-type: none"> <li>ETagListDetails in_TagListDetails: Every combination of the following four areas: <b>IO</b>: Inputs and Outputs <b>M</b>: Bit memory <b>CT</b>: Counters and Timers <b>DB</b>: Data Blocks  <b>Example: IOM</b> reads only the tags from the area Inputs / Outputs and Bit memory. The default setting is <b>IOMCTDB</b>.</li> <li>bool in_IsHMIVisibleOnly: If true, only tags marked with "HMI Visible" are read. The default setting is true.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
ERuntimeError-Code.NotEnoughMemory	More than 500000 entries are requested.	

### GetTagListStatus()

Returns the current update status of the tag list storage.

"inout\_TagListDetails" is NONE, if the list needs to be updated.

Table 7- 127 GetTagListStatus() - Native C++

Syntax	<pre>ERuntimeErrorCode GetTagListStatus(     ETagListDetails* out_TagListDetails,     bool* out_IsHMIVisibleOnly );</pre>	
Parameters	<ul style="list-style-type: none"> <li>ETagListDetails out_TagListDetails: Status of the tag list details. SRTLD_NONE when an update of the list is required.</li> <li>bool out_IsHMIVisibleOnly: If true, only tags marked with "HMI Visible" are available in the list.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.

Table 7- 128 GetTagListStatus() - .NET (C#)

Syntax	<pre>void GetTagListStatus(     out ETagListDetails out_TagListDetails,     out bool out_IsHMIVisibleOnly );</pre>	
Parameters	<ul style="list-style-type: none"> <li>out ETagListDetails out_TagListDetails: Status of the tag list details. ETagListDetails.None when an update of the list is required.</li> <li>out bool out_IsHMIVisibleOnly: If true, only tags marked with "HMI Visible" are available in the list.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.

### GetTagInfoCount()

Returns the number of entries in the tag list storage. If the function fails, the return value is 0.

Table 7- 129 GetTagInfoCount() - Native C++

Syntax	UINT32 GetTagInfoCount();
Parameters	None
Return values	Number of entries in the tag list storage.

**GetTagInfos() / TagInfos { get; }**

Returns a list of all tags.

Table 7- 130 GetTagInfos() - Native C++

Syntax	ERuntimeErrorCode GetTagInfos( UINT32 in_BufferLength, STagInfo* inout_TagInfos, UINT32* out_TagCount );	
Parameters	<ul style="list-style-type: none"> <li>• <b>UINT32 in_BufferLength:</b> The number of elements that the storage can accommodate.</li> <li>• <b>STagInfo* inout_TagInfos:</b> The user-allocated storage that accommodates the tags.</li> <li>• <b>UINT32* out_TagCount:</b> Returns the number of tags that were written to the storage.</li> </ul>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	The elements do not fit in the storage.

Table 7- 131 TagInfos { get; } - .NET (C#)

Syntax	STagInfo[] TagInfos { get; }	
Parameters	None	
Return values	An array that contains all available entries of the storage.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeErrorCode.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.InstanceNotRunning	The process of the virtual controller is not running.

**CreateConfigurationFile()**

Writes all entries from the tag list to an XML file.

Table 7- 132 CreateConfigurationFile() - Native C++

Syntax	<code>ERuntimeErrorCode CreateConfigurationFile( WCHAR* in_FullFileName );</code>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_FullFileName: Full file name of the XML file: &lt;Path&gt; + &lt;File name&gt; + &lt;File extension&gt;.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_WRONG_ARGUMENT	The file name is invalid.

Table 7- 133 CreateConfigurationFile() - .NET (C#)

Syntax	<code>void CreateConfigurationFile( string in_FullFileName );</code>	
Parameters	None	
Return values	<ul style="list-style-type: none"> <li>string in_FullFileName: File name of the XML file that is to be written to: &lt;Path&gt; + &lt;File name&gt; + &lt;File extension&gt;.</li> </ul>	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeErrorCode.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.WrongArgument	The file name is invalid.

## 7.6.5 I/O access

### 7.6.5.1 I/O access via address - Reading

#### InputArea { get; }, MarkerArea { get; }, OutputArea { get; }

Returns an interface that you use to call the following .NET functions of section "I/O access via address".

Table 7- 134 InputArea { get; } MarkerArea { get; } OutputArea { get; } - .NET (C#)

Syntax	<code>IIOArea InputArea { get; } IIOArea MarkerArea { get; } IIOArea OutputArea { get; }</code>
Parameters	None
Return values	IIOArea: The interface is used to call the "I/O access via address" functions.

#### GetAreaSize() / AreaSize { get; }

Returns the size of the area in bytes.

Table 7- 135 GetAreaSize() - Native C++

Syntax	<code>UINT32 GetAreaSize( EArea in_Area );</code>
Parameters	<ul style="list-style-type: none"> <li>EArea in_Area:  The area whose size you want to receive. Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}. See EArea (Page 286).</li> </ul>
Return values	UINT32: Size of the area in bytes. If the function was successful, the value is not equal to 0.

Table 7- 136 AreaSize { get; } - .NET (C#)

Syntax	<code>UInt32 InputArea.AreaSize { get; } UInt32 MarkerArea.AreaSize { get; } UInt32 OutputArea.AreaSize { get; }</code>
Parameters	None
Return values	UInt32: Size of the area in bytes. If the function was successful, the value is not equal to 0.

### ReadBit()

Reads an individual bit from the area.

**Note**

The function allows access to the entire storage area of the virtual controller.

**Recommendation:** Therefore, use access via the tag name and not via the address areas.

Table 7- 137 ReadBit() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadBit(     EArea in_Area,     UINT32 in_Offset,     UINT8 in_Bit,     bool* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• EArea in_Area: The area from which you want to read. Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}. See EArea (Page 286).</li> <li>• UINT32 in_Offset: The byte offset within the area. The value must be between 0 and the value that GetAreaSize() returns.</li> <li>• UINT8 in_Bit: The bit offset within the byte. The value must be between 0 and 7.</li> <li>• bool* out_Value: Returns the bit value.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	Offset or bits are invalid.
	SREC_WRONG_ARGUMENT	The area is invalid.

Table 7- 138 ReadBit() - .NET (C#)

Syntax	<pre>bool InputArea.ReadBit(     UInt32 in_Offset,     Byte in_Bit );  bool MarkerArea.ReadBit(     UInt32 in_Offset,     Byte in_Bit );  bool OutputArea.ReadBit(     UInt32 in_Offset,     Byte in_Bit );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <b>UInt32 in_Offset:</b> The byte offset within the area. The value must be between 0 and the value that <code>AreaSize</code> returns.</li> <li>• <b>Byte in_Bit:</b> The bit offset within the byte. The value must be between 0 and 7.</li> </ul>	
Return values	bool: Bit value	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeError-Code.IndexOutOfRange</code>	Offset or bits are invalid.

## ReadByte()

Reads an individual bit from the area.

### Note

The function allows access to the entire storage area of the virtual controller.

**Recommendation:** Therefore, use access via the tag name and not via the address areas.

Table 7- 139 ReadByte() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadByte(     EArea in_Area,     UINT32 in_Offset,     BYTE* out_Value);</pre>	
Parameters	<ul style="list-style-type: none"> <li>EArea in_Area: The area from which you want to read. Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}. See EArea (Page 286).</li> <li>UINT32 in_Offset: The byte offset within the area. The value must be between 0 and the value that GetAreaSize() returns.</li> <li>BYTE* out_Value: Returns the byte value.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	Offset is invalid.
SREC_WRONG_ARGUMENT	The area is invalid.	



Table 7- 140 ReadByte() - .NET (C#)

Syntax	<pre> Byte InputArea.ReadByte(     UInt32 in_Offset ); Byte MarkerArea.ReadByte(     UInt32 in_Offset ); Byte OutputArea.ReadByte(     UInt32 in_Offset ); </pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UInt32 in_Offset</code>: The byte offset within the area. The value must be between 0 and the value that <code>AreaSize</code> returns.</li> </ul>	
Return values	Byte: Byte value.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeError-Code.IndexOutOfRange</code>	Offset is invalid.

### ReadBytes()

Reads a byte array from the area.

**Note**

The function allows access to the entire storage area of the virtual controller.

**Recommendation:** Therefore, use access via the tag name and not via the address areas.

Table 7- 141 ReadByte() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadBytes (     EArea in_Area,     UINT32 in_Offset,     UINT32 in_BytesToRead,     UINT32* out_BytesRead,     BYTE inout_Values[] );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• EArea in_Area: The area from which you want to read. Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}. See EArea (Page 286).</li> <li>• UINT32 in_Offset: The byte offset within the area. The value must be between 0 and the value that GetAreaSize() returns.</li> <li>• UINT32 in_BytesToRead: Contains the size of the value storage.</li> <li>• UINT32* out_BytesRead: Returns the number of bytes that were just written to the value storage.</li> <li>• BYTE inout_Values[]: The storage for the bytes that are read from the area.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	The offset is outside the area size. No byte could be read.
	SREC_WRONG_ARGUMENT	The area is invalid.

Table 7- 142 ReadBytes() - .NET (C#)

Syntax	<pre> Byte[] InputArea.ReadBytes (     UInt32 in_Offset,     UInt32 in_BytesToRead ); Byte[] MarkerArea.ReadBytes (     UInt32 in_Offset,     UInt32 in_BytesToRead ); Byte[] OutputArea.ReadBytes (     UInt32 in_Offset,     UInt32 in_BytesToRead ); </pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UInt32 in_Offset</code>: The byte offset within the area. The value must be between 0 and the value that <code>AreaSize</code> returns.</li> <li>• <code>UInt32 in_BytesToRead</code>: The number of bytes to be read.</li> </ul>	
Return values	<code>Byte[]</code> : The read bytes.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeError-Code.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeError-Code.IndexOutOfRange</code>	The offset is outside the area size. No byte could be read.

## ReadSignals()

Reads a signal list altogether (structures and arrays). The function also takes into consideration the byte order (Endianness).

Only primitive data type signals are supported, but the function is not typical.

### Note

The function allows access to the entire storage area of the virtual controller.

**Recommendation:** Therefore, use access via the tag name and not via the address areas.

Table 7- 143 ReadSignals() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadSignals(     EArea in_Area,     SDataValueByAddress* inout_Signals,     UINT32 in_SignalCount );</pre>	
Parameters	<ul style="list-style-type: none"> <li>EArea in_Area: The area from which you want to read. Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}. See EArea (Page 286).</li> <li>SDataValueByAddress* inout_Signals: The signal list to be read. The result is stored in the structure.</li> <li>UINT32 in_SignalCount: Number of signals in the list.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	Offset or bits are invalid.
	SREC_WRONG_ARGUMENT	The area is invalid.

Table 7- 144 ReadSignals() - .NET (C#)

Syntax	<pre>void ReadSignals(     ref SDataValueByAddress[] inout_Signals );</pre>	
Parameters	<ul style="list-style-type: none"> <li>ref SDataValueByAddress[] inout_Signals:</li> </ul> <p>The signal list to be read.</p>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeError-Code.IndexOutOfRange</code>	Offset or bits are invalid.

7.6.5.2 I/O access via address - Writing

WriteBit()

Writes an individual bit to the area.

**Note**

**Data can be overwritten**

The function allows access to the entire storage area of the virtual controller.

**Recommendation:** Therefore, use access via the tag name and not via the address areas.

Table 7- 145 WriteBit() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteBit(     EArea in_Area,     UINT32 in_Offset,     UINT8 in_Bit,     bool in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• EArea in_Area: The area that is to be written. Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}. See EArea (Page 286).</li> <li>• UINT32 in_Offset: The byte offset within the area. The value must be between 0 and the value that GetAreaSize() returns.</li> <li>• UINT8 in_Bit: The bit offset within the byte. The value must be between 0 and 7.</li> <li>• bool in_Value: Bit value.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	Offset or bits are invalid.
SREC_WRONG_ARGUMENT	Area is invalid.	

Table 7- 146 WriteBit() - .NET (C#)

Syntax	<pre>void InputArea WriteBit(     UInt32 in_Offset,     Byte in_Bit,     bool in_Value ); void MarkerArea WriteBit(     UInt32 in_Offset,     Byte in_Bit,     bool in_Value ); void OutputArea WriteBit(     UInt32 in_Offset,     Byte in_Bit,     bool in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <code>UInt32 in_Offset</code>: The byte offset within the area. The value must be between 0 and the value that <code>AreaSize</code> returns.</li> <li>• <code>Byte in_Bit</code>: The bit offset within the byte. The value must be between 0 and 7.</li> <li>• <code>bool in_Value</code>: Bit value.</li> </ul>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
<code>ERuntimeError-Code.IndexOutOfRange</code>	Offset or bits are invalid.	

## WriteByte()

Writes an individual byte to the area.

---

### Note

#### Data can be overwritten

The function allows access to the entire storage area of the virtual controller.

**Recommendation:** Therefore, use access via the tag name and not via the address areas.

---

Table 7- 147 WriteByte() - Native C++

Syntax	ERuntimeErrorCode WriteByte( EArea in Area, UINT32 in Offset, BYTE in Value);	
Parameters	<ul style="list-style-type: none"> <li>EArea in_Area: The area that is to be written. Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}. See EArea (Page 286).</li> <li>UINT32 in_Offset: The byte offset within the area. The value must be between 0 and the value that GetAreaSize() returns.</li> <li>BYTE in_Value: Byte value.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	Offset is invalid.
	SREC_WRONG_ARGUMENT	Area is invalid.



Table 7- 148 WriteByte() - .NET (C#)

Syntax	<pre>void InputArea.WriteByte(     UInt32 in_Offset,     Byte in_Value );  void MarkerArea.WriteByte(     UInt32 in_Offset,     Byte in_Value );  void OutputArea.WriteByte(     UInt32 in_Offset,     Byte in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• <b>UINT32 in_Offset:</b> The byte offset within the area. The value must be between 0 and the value that <code>AreaSize</code> returns.</li> <li>• <b>BYTE in_Value:</b> Byte value.</li> </ul>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeError-Code.IndexOutOfRange</code>	Offset is invalid.

## WriteBytes()

Writes a byte array to the area.

### Note

#### Data can be overwritten

The function allows access to the entire storage area of the virtual controller.

**Recommendation:** Therefore, use access via the tag name and not via the address areas.

Table 7- 149 WriteBytes() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteBytes(     EArea in_Area,     UINT32 in_Offset,     UINT32 in_BytesToWrite,     UINT32* out_BytesWritten,     BYTE in_Values[]) ;</pre>	
Parameters	<ul style="list-style-type: none"> <li> <b>EArea in_Area:</b>                      The area that is to be written.                      Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}.                      See EArea (Page 286).                 </li> <li> <b>UINT32 in_Offset:</b>                      The byte offset within the area. The value must be between 0 and the value that GetAreaSize() returns.                 </li> <li> <b>UINT32 in_BytesToWrite:</b>                      Contains the size of the array value to be written.                 </li> <li> <b>UINT32* out_BytesWritten:</b>                      Contains the number of bytes that were just written.                 </li> <li> <b>BYTE in_Values[]:</b>                      Byte array that is to be written to the area.                 </li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	The offset is outside the area size. No byte could be written.
	SREC_WRONG_ARGUMENT	The area is invalid.

Table 7- 150 WriteBytes() - .NET (C#)

Syntax	<pre> UInt32 InputArea.WriteBytes(     UInt32 in_Offset,     Byte[] in_Values );  UInt32 InputArea.WriteBytes(     UInt32 in_Offset,     UInt32 in_BytesToWrite,     Byte[] in_Values );  UInt32 MarkerArea.WriteBytes(     UInt32 in_Offset,     Byte[] in_Values );  UInt32 MarkerArea.WriteBytes(     UInt32 in_Offset,     UInt32 in_BytesToWrite,     Byte[] in_Values );  UInt32 OutputArea.WriteBytes(     UInt32 in_Offset,     Byte[] in_Values );  UInt32 OutputArea.WriteBytes(     UInt32 in_Offset,     UInt32 in_BytesToWrite,     Byte[] in_Values ); </pre>	
Parameters	<ul style="list-style-type: none"> <li>• <b>UINT32 in_Offset:</b> The byte offset within the area. The value must be between 0 and the value that AreaSize returns.</li> <li>• <b>UInt32 in_BytesToWrite:</b> Contains the number of bytes to be written. The value must be between 1 and the size of the array value.</li> <li>• <b>BYTE in_Value:</b> Byte value.</li> </ul>	
Return values	<b>UInt32:</b> Contains the number of bytes that were just written.	
Exceptions	<i>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</i>	
	<b>Runtime error code</b>	<b>Condition</b>
	<i>ERuntimeError-Code.InterfaceRemoved</i>	The instance is not registered in Runtime Manager.
	<i>ERuntimeErrorCode.Timeout</i>	The function does not return on time.
	<i>ERuntimeError-Code.InstanceNotRunning</i>	The process of the virtual controller is not running.
<i>ERuntimeError-Code.IndexOutOfRange</i>	The offset is outside the area size. No byte could be written.	

## WriteSignals()

Writes a signal list altogether (structures and arrays). The function also takes into consideration the byte order (Endianness).

The function supports only primitive data type signals. The function is not typical, however.

### Note

#### Data can be overwritten

The function allows access to the entire storage area of the virtual controller.

**Recommendation:** Therefore, use access via the tag name and not via the address areas.

Table 7- 151 WriteSignals() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteSignals(     EArea in_Area,     SDataValueByAddress* in_Signals,     UINT32 in_SignalCount );</pre>	
Parameters	<ul style="list-style-type: none"> <li>EArea in_Area: The area that is to be written. Permissible values: {SRA_INPUT, SRA_MARKER, SRA_OUTPUT}. See EArea (Page 286).</li> <li>SDataValueByAddress* inout_Signals: The signal list to be written.</li> <li>UINT32 in_SignalCount: Number of signals in the list.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_INDEX_OUT_OF_RANGE	Offset or bits are invalid.
	SREC_WRONG_ARGUMENT	The area is invalid.

Table 7- 152 WriteSignals() - .NET (C#)

Syntax	<pre>void InputArea.WriteSignals(     SDataValueByAddress[] in_Signals ); void MarkerArea.WriteSignals(     SDataValueByAddress[] in_Signals ); void OutputArea.WriteSignals(     SDataValueByAddress[] in_Signals );</pre>	
Parameters	<ul style="list-style-type: none"> <li>SDataValueByAddress[] in_Signals: The signal list to be written.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
ERuntimeError-Code.IndexOutOfRange	Offset or bits are invalid.	

### 7.6.5.3 I/O access via tag name - Reading

#### Read()

Reads the value of a PLC tag.

Table 7- 153 Read() - Native C++

Syntax	<pre>ERuntimeErrorCode Read(     WCHAR* in_Tag,     SDataValue* inout_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>SDataValue* inout_Value: Contains the value and the expected type of the PLC tag. If the expected type is UNSPECIFIC, it is set to the stored type when the function was successful. The STRUCT type is not supported. Structures and arrays can be read altogether with ReadSignals().</li> </ul>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
	SREC_NOT_UP_TO_DATE	The stored tag list must be updated.

Table 7- 154 Read() - .NET (C#)

Syntax	SDataValue Read( string in_Tag )	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be read.</li> </ul>	
Return values	SDataValue: Contains the value and the type of the PLC tag.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types.
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### ReadBool()

Reads the value of a PLC tag.

Table 7- 155 ReadBool() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadBool(     WCHAR* in_Tag,     bool* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>• bool* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	



Table 7- 156 ReadBool() - .NET (C#)

Syntax	<code>bool ReadBool( string in_Tag )</code>	
Parameters	<ul style="list-style-type: none"> <li><code>string in_Tag</code>: The name of the PLC tag that is to be read.</li> </ul>	
Return values	<code>bool</code> : Contains the value of the PLC tag.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError- Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError- Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

**ReadInt8()**

Reads the value of a PLC tag.

Table 7- 157 ReadInt8() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadInt8(     WCHAR* in_Tag,     INT8* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>• INT8* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 158 ReadInt8() - .NET (C#)

Syntax	<code>Int8_ReadInt8( string in_Tag )</code>	
Parameters	<ul style="list-style-type: none"> <li><code>string in_Tag</code>: The name of the PLC tag that is to be read.</li> </ul>	
Return values	<code>Int8</code> : Contains the value and the type of the PLC tag.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

### ReadInt16()

Reads the value of a PLC tag.

Table 7- 159 ReadInt16() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadInt16(     WCHAR* in_Tag,     INT16* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>INT16* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 160 ReadInt16() - .NET (C#)

Syntax	<code>Int16 ReadInt16( string in_Tag )</code>	
Parameters	<ul style="list-style-type: none"> <li><code>string in_Tag</code>: The name of the PLC tag that is to be read.</li> </ul>	
Return values	<code>Int16</code> : Contains the value and the type of the PLC tag.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

### ReadInt32()

Reads the value of a PLC tag.

Table 7- 161 ReadInt32() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadInt32(     WCHAR* in_Tag,     INT32* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>INT32* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 162 ReadInt32() - .NET (C#)

Syntax	<code>Int32 ReadInt32( string in_Tag )</code>	
Parameters	<ul style="list-style-type: none"> <li><code>string in_Tag</code>: The name of the PLC tag that is to be read.</li> </ul>	
Return values	<code>Int32</code> : Contains the value and the type of the PLC tag.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

### ReadInt64()

Reads the value of a PLC tag.

Table 7- 163 ReadInt64() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadInt64(     WCHAR* in_Tag,     INT64* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>INT64* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	



Table 7- 164 ReadInt64() - .NET (C#)

Syntax	<code>Int64 ReadInt64( string in_Tag )</code>	
Parameters	<ul style="list-style-type: none"> <li><code>string in_Tag</code>: The name of the PLC tag that is to be read.</li> </ul>	
Return values	<code>Int64</code> : Contains the value and the type of the PLC tag.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

### ReadUInt8()

Reads the value of a PLC tag.

Table 7- 165 ReadUInt8() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadUInt8(     WCHAR* in_Tag,     UINT8* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>UINT8* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 166 ReadUInt8() - .NET (C#)

Syntax	<pre>UInt8 ReadUInt8(     string in_Tag )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be read.</li> </ul>	
Return values	UInt8: Contains the value and the type of the PLC tag.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### ReadUInt16()

Reads the value of a PLC tag.

Table 7- 167 ReadUInt16() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadUInt16(     WCHAR* in_Tag,     UINT16* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>UINT16* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 168 ReadUInt16() - .NET (C#)

Syntax	<pre>UInt16 ReadUInt16(     string in_Tag )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be read.</li> </ul>	
Return values	UInt16: Contains the value and the type of the PLC tag.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### ReadUInt32()

Reads the value of a PLC tag.

Table 7- 169 ReadUInt32() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadUInt32(     WCHAR* in_Tag,     UINT32* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>UINT32* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 170 ReadUInt32() - .NET (C#)

Syntax	<pre>UInt32 ReadUInt32(     string in_Tag )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be read.</li> </ul>	
Return values	UInt32: Contains the value and the type of the PLC tag.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### ReadUInt64()

Reads the value of a PLC tag.

Table 7- 171 ReadInt64() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadUInt64(     WCHAR* in_Tag,     UInt64* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>UInt64* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	



Table 7- 172 ReadUInt64() - .NET (C#)

Syntax	<pre>UInt64 ReadUInt64(     string in_Tag )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be read.</li> </ul>	
Return values	<b>UInt64:</b> Contains the value and the type of the PLC tag.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

**ReadFloat()**

Reads the value of a PLC tag.

Table 7- 173 ReadFloat() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadFloat(     WCHAR* in_Tag,     float* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>float* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 174 ReadFloat() - .NET (C#)

Syntax	<code>float ReadFloat( string in_Tag )</code>	
Parameters	<ul style="list-style-type: none"> <li><code>string in_Tag</code>: The name of the PLC tag that is to be read.</li> </ul>	
Return values	<code>float</code> : Contains the value and the type of the PLC tag.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

### ReadDouble()

Reads the value of a PLC tag.

Table 7- 175 ReadDouble() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadDouble(     WCHAR* in_Tag,     double* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>double* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 176 ReadDouble() - .NET (C#)

Syntax	<code>double ReadDouble( string in_Tag )</code>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be read.</li> </ul>	
Return values	double: Contains the value and the type of the PLC tag.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### ReadChar()

Reads the value of a PLC tag.

Table 7- 177 ReadChar() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadChar(     WCHAR* in_Tag,     char* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>char* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 178 ReadChar() - .NET (C#)

Syntax	sbyte ReadChar( string in_Tag )	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be read.</li> </ul>	
Return values	sbyte: Contains the value and the type of the PLC tag.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### ReadWChar()

Reads the value of a PLC tag.

Table 7- 179 ReadWChar() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadWChar(     WCHAR* in_Tag,     WCHAR* out_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• WCHAR* in_Tag: The name of the PLC tag that is to be read.</li> <li>• WCHAR* out_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	



Table 7- 180 ReadWChar() - .NET (C#)

Syntax	<code>char ReadWChar( string in_Tag )</code>	
Parameters	<ul style="list-style-type: none"> <li><code>string in_Tag</code>: The name of the PLC tag that is to be read.</li> </ul>	
Return values	<code>char</code> : Contains the value and the type of the PLC tag.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	<b>Runtime error code</b>	<b>Condition</b>
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

### ReadSignals()

Reads multiple signals within an API call.

*Reads multiple signals within a single API call.*

Table 7- 181 ReadSignals() - Native C++

Syntax	<pre>ERuntimeErrorCode ReadSignals(     SDataValueByName* inout_Signals,     UINT32 in_SignalCount );</pre>	
Parameters	<ul style="list-style-type: none"> <li>SDataValueByName* inout_Signals: Contains the name, the value and the expected type of the PLC tag. If the expected type is UNSPECIFIC, it is set to the stored type when the function was successful. The STRUCT type is not supported.</li> <li>UINT32 in_SignalCount: The number of signals to be read.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
	SREC_NOT_UP_TO_DATE	The stored tag list must be updated.

Table 7- 182 ReadSignals() - .NET (C#)

Syntax	void ReadSignals( ref SDataValueByName[] inout_Signals )	
Parameters	<ul style="list-style-type: none"> <li>ref SDataValueByName[] inout_Signals:</li> </ul> <p>Contains the name, the value and the expected type of the PLC tag. If the expected type is UNSPECIFIC, it is set to the stored type when the function was successful. The STRUCT type is not supported.</p>	
Return values	SDataValue: Contains the value and the type of the PLC tag.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeErrorCode.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
	ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.

### 7.6.5.4 I/O access via tag name - Writing

#### Write()

Writes the value of a PLC tag.

Table 7- 183 Write() - Native C++

Syntax	<pre>ERuntimeErrorCode Write(     WCHAR* in_Tag,     SDataValue* in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>SDataValue* in_Value: Contains the value and the expected type of the PLC tag. The UNSPECIFIC and STRUCT types are not supported. Structures and arrays can be written altogether with writeSignals().</li> </ul>	
Return values	<b>Runtime error code</b>	<b>Condition</b>
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
	SREC_NOT_UP_TO_DATE	The stored tag list must be updated.
	SREC_WRONG_ARGUMENT	The expected type is UNSPECIFIC.

Table 7- 184 Write() - .NET (C#)

Syntax	<pre>void Write(     string in_Tag     SDataValue in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>SDataValue in_Value: Contains the value and the expected type of the PLC tag. The UNSPECIFIC and STRUCT types are not supported. Structures and arrays can be written altogether with WriteSignals().</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
	ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.
ERuntimeErrorCode.WrongArgument	The expected type is UNSPECIFIC.	

### WriteBool()

Writes the value of a PLC tag.

Table 7- 185 WriteBool() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteBool(     WCHAR* in_Tag,     bool in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>bool in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 186 WriteBool() - .NET (C#)

Syntax	<pre>void WriteBool(     string in_Tag     bool in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>bool in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types.
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

**WriteInt8()**

Writes the value of a PLC tag.

Table 7- 187 WriteInt8() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteInt8(     WCHAR* in_Tag,     INT8 in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>INT8 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	



Table 7- 188 WriteInt8() - .NET (C#)

Syntax	<pre>void WriteInt8(     string in_Tag     Int8 in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>Int8 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### WriteInt16()

Writes the value of a PLC tag.

Table 7- 189 WriteInt16() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteInt16(     WCHAR* in_Tag,     INT16 in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>INT16 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 190 WriteInt16() - .NET (C#)

Syntax	<pre>void WriteInt16(   string in_Tag   Int16 in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>Int16 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### WriteInt32()

Writes the value of a PLC tag.

Table 7- 191 WriteInt32() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteInt32(     WCHAR* in_Tag,     INT32 in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>INT32 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 192 WriteInt32() - .NET (C#)

Syntax	<pre>void WriteInt32(     string in_Tag     Int32 in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>Int32 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

**WriteInt64()**

Writes the value of a PLC tag.

Table 7- 193 WriteInt64() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteInt64(     WCHAR* in_Tag,     INT64 in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>• INT64 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 194 WriteInt64() - .NET (C#)

Syntax	<pre>void WriteInt64(     string in_Tag     Int64 in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>Int64 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

**WriteInt8()**

Writes the value of a PLC tag.

Table 7- 195 WriteUInt8() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteUInt8(     WCHAR* in_Tag,     UINT8 in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>• UINT8 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	



Table 7- 196 WriteUInt8() - .NET (C#)

Syntax	<pre>void WriteUInt8(     string in_Tag     UInt8 in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>UInt8 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### WriteUInt16()

Reads the value of a PLC tag.

Table 7- 197 WriteUInt16() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteUInt16(     WCHAR* in_Tag,     UINT16 in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>UINT16 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 198 WriteUInt16() - .NET (C#)

Syntax	<pre>void WriteUInt16(     string in_Tag     UInt16 in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>UInt16 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### WriteUInt32()

Writes the value of a PLC tag.

Table 7- 199 WriteUInt32() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteUInt32(     WCHAR* in_Tag,     UINT32 in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>UINT32 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 200 WriteUInt32() - .NET (C#)

Syntax	<pre>void WriteUInt32(     string in_Tag     UInt32 in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>UInt32 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### WriteUInt64()

Writes the value of a PLC tag.

Table 7- 201 WriteUInt64() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteUInt64(     WCHAR* in_Tag,     UINT64 in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>UINT64 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 202 WriteUInt64() - .NET (C#)

Syntax	<pre>void WriteUInt64(     string in_Tag     UInt64 in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>UInt64 in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### WriteFloat()

Writes the value of a PLC tag.

Table 7- 203 WriteFloat() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteFloat(     WCHAR* in_Tag,     float in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>• float in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	



Table 7- 204 WriteFloat() - .NET (C#)

Syntax	<pre>void WriteFloat(     string in_Tag     float in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>float in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### WriteDouble()

Writes the value of a PLC tag.

Table 7- 205 WriteDouble() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteDouble(     WCHAR* in_Tag,     double in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>double in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 206 WriteDouble() - .NET (C#)

Syntax	<pre>void WriteDouble(     string in_Tag     double in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>double in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

## WriteChar()

Writes the value of a PLC tag.

Table 7- 207 WriteChar() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteChar(     WCHAR* in_Tag,     char in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>char in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 208 WriteChar() - .NET (C#)

Syntax	<pre>void WriteChar(     string in_Tag     sbyte in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>sbyte in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.DoesNotExist</code>	The entry does not exist in the stored tag list.
	<code>ERuntimeErrorCode.NotSupported</code>	Access to entire structures or arrays is not supported.
	<code>ERuntimeErrorCode.TypeMismatch</code>	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
<code>ERuntimeErrorCode.NotUpToDate</code>	The stored tag list must be updated.	

### WriteWChar()

Writes the value of a PLC tag.

Table 7- 209 WriteWChar() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteWChar(     WCHAR* in_Tag,     WCHAR in_Value );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_Tag: The name of the PLC tag that is to be written.</li> <li>WCHAR in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
SREC_NOT_UP_TO_DATE	The stored tag list must be updated.	

Table 7- 210 WriteWChar() - .NET (C#)

Syntax	<pre>void WriteWChar(     string in_Tag     char in_Value )</pre>	
Parameters	<ul style="list-style-type: none"> <li>string in_Tag: The name of the PLC tag that is to be written.</li> <li>char in_Value: Contains the value of the PLC tag.</li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.	

### WriteSignals()

Writes multiple signals within an API call.

Table 7- 211 WriteSignals() - Native C++

Syntax	<pre>ERuntimeErrorCode WriteSignals(     SDataValueByName* inout_Signals,     UINT32 in_SignalCount );</pre>	
Parameters	<ul style="list-style-type: none"> <li>SDataValueByName* inout_Signals: Contains the name, the value and the expected type of the PLC tag. The UNSPECIFIC and STRUCT types are not supported.</li> <li>UINT32 in_SignalCount: Number of signals.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INSTANCE_NOT_RUNNING	The process of the virtual controller is not running.
	SREC_DOES_NOT_EXIST	The entry does not exist in the stored tag list.
	SREC_NOT_SUPPORTED	Access to entire structures or arrays is not supported.
	SREC_TYPE_MISMATCH	The expected type does not match the stored type. See Compatible primitive data types (Page 294).
	SREC_NOT_UP_TO_DATE	The stored tag list must be updated.
SREC_WRONG_ARGUMENT	The expected type is UNSPECIFIC.	



Table 7- 212 WriteSignals() - .NET (C#)

Syntax	<pre>void WriteSignals(     SDataValueByName[] in_Signals )</pre>	
Parameters	<ul style="list-style-type: none"> <li>SDataValueByName:                      Contains the name, the value and the expected type of the PLC tag. The UNSPECIFIC and STRUCT types are not supported.                 </li> </ul>	
Return values	None	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeError-Code.InstanceNotRunning	The process of the virtual controller is not running.
	ERuntimeErrorCode.DoesNotExist	The entry does not exist in the stored tag list.
	ERuntimeErrorCode.NotSupported	Access to entire structures or arrays is not supported.
	ERuntimeErrorCode.TypeMismatch	The expected type does not match the stored type. See Compatible primitive data types.
	ERuntimeErrorCode.NotUpToDate	The stored tag list must be updated.
ERuntimeErrorCode.WrongArgument	The expected type is UNSPECIFIC.	

## 7.6.6 Settings for the virtual time

### GetSystemTime()

Returns the virtual system time of the virtual controller. Returns an empty structure when the function fails.

Table 7- 213 GetSystemTime() - Native C++

Syntax	<code>SYSTEMTIME GetSystemTime();</code>
Parameters	None
Return values	<code>SYSTEMTIME</code> : System time of the virtual controller.

### SetSystemTime()

Sets the virtual system time of the virtual controller. A system time between "Jan 1 1970 00:00:00:000" and "Dec 31 2200 23:59:59:999" is valid.

Table 7- 214 SetSystemTime() - Native C++

Syntax	<code>ERuntimeErrorCode SetSystemTime( SYSTEMTIME in_SystemTime );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>SYSTEMTIME in_SystemTime</code>: System time that is to be set for the virtual controller.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The function does not return on time.
	<code>SREC_WRONG_ARGUMENT</code>	The value is outside the limits.

### SystemTime { get; set; }

Sets or returns the virtual system time of the virtual controller. A system time between "Jan 1 1970 00:00:00:000" and "Dec 31 2200 23:59:59:999" is valid.

Table 7- 215 SystemTime { get; set; } - .NET (C#)

Syntax	<code>DateTime SystemTime { get; set; }</code>	
Parameters	None	
Return values	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeErrorCode.WrongArgument</code>	The value is outside the limits.

## GetScaleFactor()

Returns the scaling factor with which the virtual time advances.

Table 7- 216 GetScaleFactor() - Native C++

Syntax	<code>double GetScaleFactor();</code>
Parameters	None
Return values	<code>double</code> : Scaling factor of the virtual time.

## SetScaleFactor()

Sets the scaling factor with which the virtual time advances.

Start with a small scaling factor and incrementally approach a scaling factor at which the virtual controller remains in RUN.

A value between 0.01 and 100 is valid. The default setting is 1.

- If the value is less than 1, the virtual time of the virtual controller runs X-times slower than the real time.
- If the value is greater than 1, the virtual time of the virtual controller runs X-times faster than the real time.

A change in the value during runtime only takes effect at the cycle control point.

Table 7- 217 SetScaleFactor() - Native C++

Syntax	<code>ERuntimeErrorCode SetScaleFactor ( double in_Value );</code>	
Parameters	<ul style="list-style-type: none"> <li>• <code>double in_Value</code>: Scaling factor of the virtual time.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The instance is not registered in Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_WRONG_ARGUMENT	The value is outside the limits.

**ScaleFactor { get; set; }**

Sets or returns the scaling factor with which the virtual time advances.

Start with a small scaling factor and incrementally approach a scaling factor at which the virtual controller remains in RUN.

A value between 0.01 and 100 is valid. The default setting is 1.

- If the value is less than 1, the virtual time of the virtual controller runs X-times slower than the real time.
- If the value is greater than 1, the virtual time of the virtual controller runs X-times faster than the real time.

A change in the value during runtime only takes effect at the cycle control point.

Table 7- 218 ScaleFactor { get; set; } - .NET (C#)

Syntax	double ScaleFactor { get; set; }	
Parameters	None	
Return values	double: Scaling factor of the virtual time.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeErrorCode.InterfaceRemoved	The instance is not registered in Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.WrongArgument	The value is outside the limits.

## 7.6.7 Cycle control

### GetOperatingMode()

Returns the operating mode of the virtual controller.

Table 7- 219 GetOperatingMode() - Native C++

Syntax	<code>EOperatingMode GetOperatingMode();</code>
Parameters	None
Return values	<code>EOperatingMode</code> : Operating mode of the virtual controller

### SetOperatingMode()

Sets the operating mode of the virtual controller.

A change in the value during runtime only takes effect at the cycle control point.

Table 7- 220 SetOperatingMode() - Native C++

Syntax	<code>void SetOperatingMode( EOperatingMode in_OperatingMode );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>EOperatingMode in_OperatingMode</code>: Operating mode of the virtual controller</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The function does not return on time.

### OperatingMode { get; set; }

Returns or sets the operating mode of the virtual controller.

A change in the value during runtime only takes effect at the cycle control point.

Table 7- 221 OperatingMode { get; set; } - .NET (C#)

Syntax	<code>EOperatingMode OperatingMode { get; set; }</code>	
Parameters	None	
Return values	<code>EOperatingMode</code> : Operating mode of the virtual controller	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError- Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.

### SetAlwaysSendOnEndOfCycleEnabled()

Sets the `AlwaysSendOnEndOfCycle mode`. If the mode is set, the `OnEndOfCycle` event is triggered after every cycle end for each operating mode. If the event is also to be received in `Default` operating mode, set the return value to `true`. See `OnEndOfCycle` (Page 232).

Table 7- 222 SetAlwaysSendOnEndOfCycleEnabled() - Native C++

Syntax	<code>ERuntimeErrorCode SetAlwaysSendOnEndOfCycleEnabled( bool in_Enable );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>bool in_Enable</code>: If <code>true</code>, the <code>OnEndOfCycle</code> event is triggered after every cycle.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The function does not return on time.

### IsAlwaysSendOnEndOfCycleEnabled()

Returns the `AlwaysSendOnEndOfCycle mode`. When the function fails, the return value is `false`.

Table 7- 223 IsAlwaysSendOnEndOfCycleEnabled() - Native C++

Syntax	<code>bool IsAlwaysSendOnEndOfCycleEnabled();</code>
Parameters	None
Return values	<ul style="list-style-type: none"> <li><code>false</code>: The event is not triggered (unless the Sync-Freeze mode is active).</li> <li><code>true</code>: The event is triggered after every cycle.</li> </ul>

### IsAlwaysSendOnEndOfCycleEnabled { get; set; }

Returns or sets the `AlwaysSendOnEndOfCycle mode`. If the mode is set, the `OnEndOfCycle` event is triggered after every cycle end for each operating mode. If the event is also to be received in `Default` operating mode, set the return value to `true`. See `OnEndOfCycle` (Page 232).

Table 7- 224 IsAlwaysSendOnEndOfCycleEnabled { get; set; } - .NET (C#)

Syntax	<code>bool IsAlwaysSendOnEndOfCycleEnabled { get; set; }</code>	
Parameters	None	
Return values	<ul style="list-style-type: none"> <li><code>false</code>: The event is not triggered (unless the Sync-Freeze mode is active).</li> <li><code>true</code>: The event is triggered after every cycle.</li> </ul>	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.

**GetOverwrittenMinimalCycleTime\_ns()**

Returns the overwritten minimum cycle time (in nanoseconds) that is used in the `ExtendedSingleStep` operating mode.

Table 7- 225 `GetOverwrittenMinimalCycleTime_ns()` - Native C++

Syntax	<code>INT64 GetOverwrittenMinimalCycleTime_ns();</code>
Parameters	None
Return values	<code>INT64</code> : The overwritten minimum cycle time in nanoseconds.

**SetOverwrittenMinimalCycleTime\_ns()**

Sets the overwritten minimum cycle time (in nanoseconds) that is used in the `ExtendedSingleStep` operating mode.

A value between 0 and 6000000000 is valid. The default setting is 100 ms.

A change in the value during runtime only takes effect at the cycle control point.

Table 7- 226 `SetOverwrittenMinimalCycleTime_ns()` - Native C++

Syntax	<code>ERuntimeErrorCode SetOverwrittenMinimalCycleTime_ns( INT64 in_CycleTime_ns );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>INT64 in_CycleTime_ns</code>: The overwritten minimum cycle time in nanoseconds.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The function does not return on time.
	<code>SREC_WRONG_ARGUMENT</code>	The value is outside the limits.

**OverwrittenMinimalCycleTime\_ns { get; set; }**

Returns or sets the overwritten minimum cycle time in nanoseconds that is used in the `ExtendedSingleStep` operating mode.

A value between 0 and 6000000000 is valid. The default setting is 100 ms.

A change in the value during runtime only takes effect at the cycle control point.

Table 7- 227 OverwrittenMinimalCycleTime\_ns { get; set; } - .NET (C#)

Syntax	<code>Int64 OverwrittenMinimalCycleTime_ns { get; set; }</code>	
Parameters	None	
Return values	<code>Int64</code> : The overwritten minimum cycle time in nanoseconds.	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeErrorCode.WrongArgument</code>	The value is outside the limits.



## RunNextCycle()

If the virtual controller is being operated in `SingleStep` or `ExtendedSingleStep` operating mode, it is stopped at the cycle control point (Freeze state). The `RunNextCycle()` function is used to trigger the next cycle.

Table 7- 228 RunNextCycle() - Native C++

Syntax	<code>ERuntimeErrorCode RunNextCycle();</code>	
Parameters	None	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The function does not return on time.
	<code>SREC_INSTANCE_NOT_RUNNING</code>	The process of the virtual controller is not running.

Table 7- 229 RunNextCycle() - .NET (C#)

Syntax	<code>void RunNextCycle();</code>	
Parameters	None	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeErrorCode.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeErrorCode.InstanceNotRunning</code>	The process of the virtual controller is not running.

### StartProcessing()

If the virtual controller is running in `TimespanSynchronized` operating mode, it is stopped at the cycle control point (Freeze state). The `StartProcessing()` function wakes up the virtual controller from the Freeze state. The virtual controller will now run for at least the requested time before it changes to Freeze state (stops) at the next cycle control point.

Table 7- 230 StartProcessing() - Native C++

Syntax	<code>ERuntimeErrorCode StartProcessing( INT64 in_MinimalTimeToRun_ns );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>INT64 in_CycleTime_ns</code>: The minimum virtual time (in nanoseconds) that the virtual controller runs before it changes to Freeze state.</li> </ul>	
Return values	Runtime error code	Condition
	<code>SREC_OK</code>	The function is successful.
	<code>SREC_INTERFACE_REMOVED</code>	The instance is not registered in Runtime Manager.
	<code>SREC_TIMEOUT</code>	The function does not return on time.
	<code>SREC_INSTANCE_NOT_RUNNING</code>	The process of the virtual controller is not running.
	<code>SREC_WRONG_ARGUMENT</code>	The value is less than 0.

Table 7- 231 StartProcessing() - .NET (C#)

Syntax	<code>void StartProcessing( Int64 in_MinimalTimeToRun_ns );</code>	
Parameters	<ul style="list-style-type: none"> <li><code>Int64 in_CycleTime_ns</code>: The minimum virtual time (in nanoseconds) that the virtual controller runs before it changes to Freeze state.</li> </ul>	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The instance is not registered in Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.
	<code>ERuntimeError-Code.InstanceNotRunning</code>	The process of the virtual controller is not running.
	<code>ERuntimeErrorCode.WrongArgument</code>	The value is less than 0.

### Additional information

For more information, please refer to the section titled Time response (Page 58), Stopping the simulation (Page 61)

**See also**

OnOperatingStateChanged (Page 228)

**7.6.8 Events****Events for Instances**

The following events are differentiated in the API:

Table 7- 232 Events for the Instances interface

Event	Cause
OnOperatingStateChanged (Page 228)	This event is always triggered when the operating state of the virtual controller has changed.
OnEndOfCycle (Page 232)	This event is triggered when the virtual controller reaches the end of the main cycle. If the virtual controller is being operated in Default operating mode, the <code>AlwaysSendOnEndOfCycle</code> flag must be set to receive the event. See <code>AlwaysSendOnEndOfCycle</code> (Page 221).
OnConfigurationChanging (Page 234)	This event is triggered when changing of the configuration of the virtual controller starts. <ul style="list-style-type: none"> <li>• During power up from the Virtual SIMATIC Memory Card</li> <li>• At the start of a download</li> </ul> When this event is triggered, the stored tag list is reset.
OnConfigurationChanged (Page 237)	This event is triggered when the configuration of the virtual controller has changed: <ul style="list-style-type: none"> <li>• After power up from the Virtual SIMATIC Memory Card</li> <li>• At the end of a download</li> <li>• When the IP address changes</li> </ul>
OnLedChanged (Page 239)	This event is triggered when the LED display of the virtual controller has changed.

### 7.6.8.1 OnOperatingStateChanged

#### OnOperatingStateChanged

Registers or unregisters an event handler method.

Table 7- 233 OnOperatingStateChanged - .NET (C#)

Syntax	<code>event Delegate_II_EREC_DT_EOS_EOS OnOperatingStateChanged;</code>
Parameters	None. See Delegate_II_EREC_DT_EOS_EOS (Page 266).
Return values	None
Exceptions	None
Note	The event handler method runs in a separate thread.

#### RegisterOnOperatingStateChangedCallback()

When the event occurs, the registered callback function is called. Only one callback function can be registered for the event. Registering a new callback function causes the previous callback function to be deleted.

Table 7- 234 RegisterOnOperatingStateChangedCallback() - Native C++

Syntax	<code>void RegisterOnOperatingStateChangedCallback( EventCallback_II_SREC_ST_SROS_SROS in_CallbackFunction );</code>
Parameters	<ul style="list-style-type: none"> <li>EventCallback_II_SREC_ST_SROS_SROS in_CallbackFunction: A callback function that subscribes to the event. See EventCallback_II_SREC_ST_SROS_SROS (Page 260).</li> </ul>
Return values	None
Note	The callback function runs in a separate thread.

**RegisterOnOperatingStateChangedEvent()**

When the event occurs, the registered event object is set to the signaled state. Only one event object can be registered for the event. Registering a new event object causes the previous event object to be deleted.

Table 7- 235 RegisterOnOperatingStateChangedEvent() - Native C++

Syntax	<pre>void RegisterOnOperatingStateChangedEvent(); void RegisterOnOperatingStateChangedEvent( HANDLE* in_Event );</pre>
Parameters	<ul style="list-style-type: none"> <li>• None: An internal event object is registered.</li> <li>• HANDLE* in_Event: A handle for a user-specific event object. The event object is registered.</li> </ul>
Return values	None
Example C++	<pre>// Thread 1 ----- ISimulationRuntimeManager * api = ZERO; ERuntimeErrorCode result = Initialize(&amp;api);  IInstance* psa = ZERO; if (result == SREC_OK) { result = api-&gt;RegisterInstance(&amp;psa); }  // Register the internal event object psa-&gt;RegisterOnOperatingStateChangedEvent();  // Thread 2 ----- while (condition) { // Wait for the event to be set (timeout after 10s) bool isEventSet = psa- &gt;WaitForOnOperatingStateChangedEvent(10000); if (isEventSet) { // Do Something ... } }</pre>

Example C++	<pre> // Thread 1 ----- ISimulationRuntimeManager * api = ZERO; ERuntimeErrorCode result = Initialize(&amp;api);  IInstance* psa = ZERO; if (result == SREC_OK) {     result = api-&gt;RegisterInstance(&amp;psa); }  // Create an event object HANDLE eventHandle = CreateEvent(ZERO, FALSE, FALSE, ZERO);  // Register the user created event object psa-&gt;<b>RegisterOnOperatingStateChangedEvent</b>(&amp;eventHandle);  // Do Something ... // Clean up the handle CloseHandle(eventHandle);  // Thread 2 ----- while (condition) {     // Wait for the event to be set //OR:     WaitForSingleObject(eventHandle, INFINITE); //psa-     &gt;WaitForOnOperatingStateChangedEvent();      // Do Something     ... } </pre>
-------------	--

**UnregisterOnOperatingStateChangedCallback()**

Unregisters the callback function. When the event occurs, no callback function is called.

Table 7- 236 UnregisterOnOperatingStateChangedCallback() - Native C++

Syntax	void UnregisterOnOperatingStateChangedCallback();
Parameters	None
Return values	None

## UnregisterOnOperatingStateChangedEvent()

Unregisters the event object.

Table 7- 237 UnregisterOnOperatingStateChangedEvent() - Native C++

Syntax	<code>void UnregisterOnOperatingStateChangedEvent();</code>
Parameters	None
Return values	None

Table 7- 238 UnregisterOnOperatingStateChangedEvent() - .NET (C#)

Syntax	<code>void UnregisterOnOperatingStateChangedEvent();</code>
Parameters	None
Return values	None

## WaitForOnOperatingStateChangedEvent()

The function blocks the program until the registered event object is in the signaled state or the timeout interval is exceeded.

Table 7- 239 WaitForOnOperatingStateChangedEvent() - Native C++

Syntax	<code>bool WaitForOnOperatingStateChangedEvent(); bool WaitForOnOperatingStateChangedEvent(   UINT32 in_Time_ms );</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to <code>INFINITE</code>.</li> <li><code>UINT32 in_Time_ms</code>: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li><code>true</code>: If the event object was set to the signaled state.</li> <li><code>false</code>: If no event was received during the defined time limit.</li> </ul>

Table 7- 240 WaitForOnOperatingStateChangedEvent() - .NET (C#)

Syntax	<code>bool WaitForOnOperatingStateChangedEvent(); bool WaitForOnOperatingStateChangedEvent(   UInt32 in_Time_ms );</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to <code>INFINITE</code>.</li> <li><code>UInt32 in_Time_ms</code>: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li><code>true</code>: If the event object was set to the signaled state.</li> <li><code>false</code>: If no event was received during the defined time limit.</li> </ul>

### 7.6.8.2 OnEndOfCycle

#### OnEndOfCycle

Registers or unregisters an event handler method.

Table 7- 241 OnEndOfCycle - .NET (C#)

Syntax	<code>event Delegate_II_EREC_DT OnEndOfCycle;</code>
Parameters	None. See Delegate_II_EREC_DT (Page 265).
Return values	None
Exceptions	None
Note	The event handler method runs in a separate thread.

#### RegisterOnEndOfCycleCallback()

When the event occurs, the registered callback function is called. Only one callback function can be registered for the event. Registering a new callback function causes the previous callback function to be deleted.

Table 7- 242 RegisterOnEndOfCycleCallback() - Native C++

Syntax	<code>void RegisterOnEndOfCycleCallback( EventCallback_II_SREC_ST in_CallbackFunction );</code>
Parameters	<ul style="list-style-type: none"> <li>EventCallback_II_SREC_ST in_CallbackFunction: A callback function that subscribes to an event. See EventCallback_II_SREC_ST (Page 259).</li> </ul>
Return values	None
Note	The callback function runs in a separate thread.

#### RegisterOnEndOfCycleEvent()

When the event occurs, the registered event object is set to the signaled state. Only one event object can be registered for the event. Registering a new event object causes the previous event object to be deleted.

Table 7- 243 RegisterOnEndOfCycleEvent() - Native C++

Syntax	<code>void RegisterOnEndOfCycleEvent(); void RegisterOnEndOfCycleEvent( HANDLE* in_Event );</code>
Parameters	<ul style="list-style-type: none"> <li>None: An internal event object is registered.</li> <li>HANDLE* in_Event: A handle for a user-specific event object. The event object is registered.</li> </ul>
Return values	None



**UnregisterOnEndOfCycleCallback()**

Unregisters the callback function. When the event occurs, no callback function is called.

Table 7- 244 UnregisterOnEndOfCycleCallback() - Native C++

Syntax	<code>void UnregisterOnEndOfCycleCallback();</code>
Parameters	None
Return values	None

**UnregisterOnEndOfCycleEvent()**

Unregisters the event object.

Table 7- 245 RegisterOnEndOfCycleEvent() - Native C++

Syntax	<code>void UnregisterOnEndOfCycleEvent();</code>
Parameters	None
Return values	None

Table 7- 246 UnregisterOnEndOfCycleEvent() - .NET (C#)

Syntax	<code>void UnregisterOnEndOfCycleEvent();</code>
Parameters	None
Return values	None

### WaitForOnEndOfCycleEvent()

The function blocks the program until the registered event object is in the signaled state or the timeout interval is exceeded.

Table 7- 247 WaitForOnEndOfCycleEvent() - Native C++

Syntax	<pre>bool WaitForOnEndOfCycleEvent (); bool WaitForOnEndOfCycleEvent (     UINT32 in_Time_ms );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UINT32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

Table 7- 248 WaitForOnEndOfCycleEvent() - .NET (C#)

Syntax	<pre>bool WaitForOnEndOfCycleEvent (); bool WaitForOnEndOfCycleEvent (     UInt32 in_Time_ms );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UInt32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

### 7.6.8.3 OnConfigurationChanging

#### OnConfigurationChanging

Registers or unregisters an event handler method.

Table 7- 249 OnConfigurationChanging - .NET (C#)

Syntax	<pre>event Delegate_II_EREC_DT OnConfigurationChanging;</pre>
Parameters	None. See Delegate_II_EREC_DT (Page 265).
Return values	None
Exceptions	None
Note	The event handler method runs in a separate thread.

## RegisterOnConfigurationChangingCallback()

When the event occurs, the registered callback function is called. Only one callback function can be registered for the event. Registering a new callback function causes the previous callback function to be deleted.

Table 7- 250 RegisterOnConfigurationChangingCallback() - Native C++

Syntax	<pre>void RegisterOnConfigurationChangingCallback(     EventCallback_II_SREC_ST in_CallbackFunction );</pre>
Parameters	<ul style="list-style-type: none"> <li>EventCallback_II_SREC_ST in_CallbackFunction: A callback function that subscribes to an event. See EventCallback_II_SREC_ST (Page 259).</li> </ul>
Return values	None
Note	The callback function runs in a separate thread.

## RegisterOnConfigurationChangingEvent()

When the event occurs, the registered event object is set to the signaled state. Only one event object can be registered for the event. Registration of a new event object causes the previous event object to be deleted.

Table 7- 251 RegisterOnConfigurationChangingEvent() - Native C++

Syntax	<pre>void RegisterOnConfigurationChangingEvent(); void RegisterOnConfigurationChangingEvent(     HANDLE* in_Event );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: An internal event object is registered.</li> <li>HANDLE* in_Event: A handle for a user-specific event object. The event object is registered.</li> </ul>
Return values	None

## UnregisterOnConfigurationChangingCallback()

Unregisters the callback function. When the event occurs, no callback function is called.

Table 7- 252 UnregisterOnConfigurationChangingCallback() - Native C++

Syntax	<pre>void UnregisterOnConfigurationChangingCallback();</pre>
Parameters	None
Return values	None

### UnregisterOnConfigurationChangingEvent()

Unregisters the event object.

Table 7- 253 UnregisterOnConfigurationChangingEvent() - Native C++

Syntax	<code>void UnregisterOnConfigurationChangingEvent();</code>
Parameters	None
Return values	None

Table 7- 254 UnregisterOnConfigurationChangingEvent() - .NET (C#)

Syntax	<code>void UnregisterOnConfigurationChangingEvent();</code>
Parameters	None
Return values	None

### WaitForOnConfigurationChangingEvent()

The function blocks the program until the registered event object is in the signaled state or the timeout interval is exceeded.

Table 7- 255 WaitForOnConfigurationChangingEvent() - Native C++

Syntax	<code>bool WaitForOnConfigurationChangingEvent(); bool WaitForOnConfigurationChangingEvent(   UINT32 in_Time_ms );</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UINT32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

Table 7- 256 WaitForOnConfigurationChangingEvent() - .NET (C#)

Syntax	<code>bool WaitForOnConfigurationChangingEvent(); bool WaitForOnConfigurationChangingEvent(   UInt32 in_Time_ms );</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UInt32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

## 7.6.8.4 OnConfigurationChanged

### OnConfigurationChanged

Registers or unregisters an event handler method.

Table 7- 257 OnConfigurationChanged - .NET (C#)

Syntax	<code>event Delegate II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32 OnConfigurationChanged;</code>
Parameters	None. See Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32 (Page 270).
Return values	None
Exceptions	None
Note	The event handler method runs in a separate thread.

### RegisterOnConfigurationChangedCallback()

When the event occurs, the registered callback function is called. Only one callback function can be registered for the event. Registering a new callback function causes the previous callback function to be deleted.

Table 7- 258 RegisterOnConfigurationChangedCallback() - Native C++

Syntax	<code>void RegisterOnConfigurationChangedCallback( _EventCallback_SRCC_UINT32_UINT32_INT32 in_CallbackFunction );</code>
Parameters	<ul style="list-style-type: none"> <li>EventCallback_SRCC_UINT32_UINT32_INT32 in_CallbackFunction: A callback function that subscribes to an event. See EventCallback_SRCC_UINT32_UINT32_INT32 (Page 263).</li> </ul>
Return values	None
Note	The callback function runs in a separate thread.

### RegisterOnConfigurationChangedEvent()

When the event occurs, the registered event object is set to the signaled state. Only one event object can be registered for the event. Registering a new event object causes the previous event object to be deleted.

Table 7- 259 RegisterOnConfigurationChangedEvent() - Native C++

Syntax	<code>void RegisterOnConfigurationChangedEvent(); void RegisterOnConfigurationChangedEvent( HANDLE* in_Event );</code>
Parameters	<ul style="list-style-type: none"> <li>None: An internal event object is registered.</li> <li>HANDLE* in_Event: A handle for a user-specific event object. The event object is registered.</li> </ul>
Return values	None

### UnregisterOnConfigurationChangedCallback()

Unregisters the callback function. When the event occurs, no callback function is called.

Table 7- 260 UnregisterOnConfigurationChangedCallback() - Native C++

Syntax	<code>void UnregisterOnConfigurationChangedCallback();</code>
Parameters	None
Return values	None

### UnregisterOnConfigurationChangedEvent()

Unregisters the event object.

Table 7- 261 UnregisterOnConfigurationChangedEvent() - Native C++

Syntax	<code>void UnregisterOnConfigurationChangedEvent();</code>
Parameters	None
Return values	None

Table 7- 262 UnregisterOnConfigurationChangedEvent() - .NET (C#)

Syntax	<code>void UnregisterOnConfigurationChangedEvent();</code>
Parameters	None
Return values	None

## WaitForOnConfigurationChangedEvent()

The function blocks the program until the registered event object is in the signaled state or the timeout interval is exceeded.

Table 7- 263 WaitForOnConfigurationChangedEvent() - Native C++

Syntax	<pre>bool WaitForOnConfigurationChangedEvent(); bool WaitForOnConfigurationChangedEvent(     UINT32 in_Time_ms );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UINT32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

Table 7- 264 WaitForOnConfigurationChangedEvent() - .NET (C#)

Syntax	<pre>bool WaitForOnConfigurationChangedEvent(); bool WaitForOnConfigurationChangedEvent(     UInt32 in_Time_ms );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UInt32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

### 7.6.8.5 OnLedChanged

#### OnLedChanged

Registers or unregisters an event handler method.

Table 7- 265 OnLedChanged - .NET (C#)

Syntax	<pre>event Delegate_II_EREC_DT_ELT_ELM OnLedChanged;</pre>
Parameters	None. See Delegate_II_EREC_DT_ELT_ELM (Page 267).
Return values	None
Exceptions	None
Note	The event handler method runs in a separate thread.

### RegisterOnLedChangedCallback()

When the event occurs, the registered callback function is called. Only one callback function can be registered for the event. Registering a new callback function causes the previous callback function to be deleted.

Table 7- 266 RegisterOnLedChangedCallback() - Native C++

Syntax	<pre>void RegisterOnLedChangedCallback(     EventCallback_II_SREC_ST_SRLT_SRLM in_CallbackFunction );</pre>
Parameters	<ul style="list-style-type: none"> <li>EventCallback_II_SREC_ST_SRLT_SRLM in_CallbackFunction: A callback function that subscribes to an event. See EventCallback_II_SREC_ST_SRLT_SRLM (Page 261).</li> </ul>
Return values	None
Note	The callback function runs in a separate thread.

### RegisterOnLedChangedEvent()

When the event occurs, the registered event object is set to the signaled state. Only one event object can be registered for the event. Registering a new event object causes the previous event object to be deleted.

Table 7- 267 RegisterOnLedChangedEvent() - Native C++

Syntax	<pre>void RegisterOnLedChangedEvent(); void RegisterOnLedChangedEvent(     HANDLE* in_Event );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: An internal event object is registered.</li> <li>HANDLE* in_Event: A handle for a user-specific event object. The event object is registered.</li> </ul>
Return values	None

### UnregisterOnLedChangedCallback()

Unregisters the callback function. When the event occurs, no callback function is called.

Table 7- 268 UnregisterOnLedChangedCallback() - Native C++

Syntax	<pre>void UnregisterOnLedChangedCallback();</pre>
Parameters	None
Return values	None



## UnregisterOnLedChangedEvent()

Unregisters the event object.

Table 7- 269 UnregisterOnLedChangedEvent() - Native C++

Syntax	<code>void UnregisterOnLedChangedEvent();</code>
Parameters	None
Return values	None

Table 7- 270 UnregisterOnLedChangedEvent() - .NET (C#)

Syntax	<code>void UnregisterOnLedChangedEvent();</code>
Parameters	None
Return values	None

## WaitForOnLedChangedEvent()

The function blocks the program until the registered event object is in the signaled state or the timeout interval is exceeded.

Table 7- 271 WaitForOnLedChangedEvent() - Native C++

Syntax	<code>bool WaitForOnLedChangedEvent(); bool WaitForOnLedChangedEvent(   UINT32 in_Time_ms );</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to <code>INFINITE</code>.</li> <li><code>UINT32 in_Time_ms</code>: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li><code>true</code>: If the event object was set to the signaled state.</li> <li><code>false</code>: If no event was received during the defined time limit.</li> </ul>

Table 7- 272 WaitForOnLedChangedEvent() - .NET (C#)

Syntax	<code>bool WaitForOnLedChangedEvent(); bool WaitForOnLedChangedEvent(   UInt32 in_Time_ms );</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to <code>INFINITE</code>.</li> <li><code>UInt32 in_Time_ms</code>: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li><code>true</code>: If the event object was set to the signaled state.</li> <li><code>false</code>: If no event was received during the defined time limit.</li> </ul>

## 7.7 API IRemoteRuntimeManager

### 7.7.1 Interfaces - Information and settings

#### Dispose()

Deletes the managed interface and unloads the native components of the user interfaces.

Table 7- 273 Dispose() - .NET (C#)

Syntax	<code>void Dispose()</code>
Parameters	None
Return values	None

#### GetVersion()

Returns the version of the remote Runtime Manager. If the function fails, version 0.0 is returned.

Table 7- 274 GetVersion() - Native C++

Syntax	<code>UINT32 GetVersion();</code>
Parameters	None
Return values	<code>UINT32: Remote Runtime Manager Version (HIWORD = Major, LOWORD = Minor)</code>

Table 7- 275 Version { get; } - .NET (C#)

Syntax	<code>UInt32 Version { get; }</code>
Parameters	None
Return values	<code>uint32: Remote Runtime Manager Version (HIWORD = Major, LOWORD = Minor)</code>

**GetIP() / IP { get; }**

Returns the IP address of the PC on which the remote Runtime Manager is running. If the function fails, the return value is 0.

Table 7- 276 GetIP() - Native C++

Syntax	UIP GetIP();
Parameters	None
Return values	UIP: Returns the IP address of the PC on which the Runtime Manager is running.

Table 7- 277 IP { get; } - .NET (C#)

Syntax	SIP IP { get; }
Parameters	None
Return values	SIP: Returns the IP address of the PC on which the Runtime Manager is running.

**GetPort() / Port { get; }**

Returns the open port of the PC on which the remote Runtime Manager is running. If the function fails, the return value is 0.

Table 7- 278 GetPort() - Native C++

Syntax	UINT16 GetPort();
Parameters	None
Return values	UINT16: Open port of the PC on which the remote Runtime Manager is running.

Table 7- 279 Port { get; } - .NET (C#)

Syntax	UInt16 Port { get; }
Parameters	None
Return values	UInt16: Open port of the PC on which the remote Runtime Manager is running.

**GetRemoteComputerName() / RemoteComputerName { get; }**

Returns the name of the PC on which the remote Runtime Manager is running.

Table 7- 280 GetRemoteComputerName() - Native C++

Syntax	<pre>ERuntimeErrorCode GetRemoteComputerName (     WCHAR* inout Name,     UINT32 in_ArYayLength );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* inout_Name: A user-allocated array for the computer name.</li> <li>UINT32 in_ArrayLength: The array length. The array should be longer than MAX_COMPUTERNAME_LENGTH.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The interface is disconnected from the remote Runtime Manager.
	SREC_INDEX_OT_OF_RANGE	The array is too small to accommodate the computer name.

Table 7- 281 RemoteComputerName { get; } - .NET (C#)

Syntax	string RemoteComputerName { get; }	
Parameters	None	
Return values	string: Name of the PC on which the remote Runtime Manager is running.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The interface is disconnected from the remote Runtime Manager.
	ERuntimeError-Code.IndexOutOfRange	The array is too small to accommodate the computer name.

**Disconnect()**

Closes the connection to the remote Runtime Manager.

**Note**

All applications that are connected to the remote Runtime Manager lose this connection.

Table 7- 282 Disconnect() - Native C++

Syntax	<code>ERuntimeErrorCode Disconnect();</code>	
Parameters	None	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The interface is disconnected from the remote Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.

Table 7- 283 Disconnect() - .NET (C#)

Syntax	<code>void Disconnect();</code>	
Parameters	None	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The interface is disconnected from the remote Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.

## 7.7.2 Simulation Runtime instances

### 7.7.2.1 Simulation Runtime instances (remote)

#### GetRegisteredInstancesCount()

Returns the number of instances that are registered in Runtime Manager. If the function fails, the return value is 0.

Table 7- 284 GetRegisteredInstancesCount() - Native C++

Syntax	UINT32 GetRegisteredInstancesCount();
Parameters	None
Return values	UINT32: Number of available instances.

#### GetRegisteredInstanceInfoAt()

Returns information about an already registered instance.

You can use the ID or name to create an interface of this instance (see `CreateInterface()`).

Table 7- 285 GetRegisteredInstanceInfoAt() - Native C++

Syntax	ERuntimeErrorCode GetRegisteredInstanceInfoAt( UINT32 in_Index, SInstanceInfo* out_InstanceInfo );	
Parameters	<ul style="list-style-type: none"> <li>UINT32 in_Index: Index of the created instance from which you want to receive the information. The index must be less than the value you receive when you call <code>GetRegisteredInstanceCount()</code>.</li> <li>SInstanceInfo* out_InstanceInfo: The information with name and ID of the instance. See <code>SInstanceInfo</code> (Page 278).</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The interface is disconnected from the remote Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_INDEX_OUT_OF_RANGE	There is no instance information for this index.

**RegisteredInstanceInfo { get; }**

Returns information about an already registered instance. You can use the ID or name of this instance to create an interface of this instance, see `CreateInterface()`.

Table 7- 286 RegisterInstanceInfo { get; } - .NET (C#)

Syntax	<code>SInstanceInfo[] RegisteredInstanceInfo { get; }</code>	
Parameters	None	
Return values	None	
Exceptions	<code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code>	
	Runtime error code	Condition
	<code>ERuntimeError-Code.InterfaceRemoved</code>	The interface is disconnected from the remote Runtime Manager.
	<code>ERuntimeErrorCode.Timeout</code>	The function does not return on time.

### RegisterInstance()

Registers a new instance of a virtual controller in Runtime Manager. Creates and returns an interface of this instance.

Table 7- 287 RegisterInstance() - Native C++

Syntax	<pre>ERuntimeErrorCode RegisterInstance(     IInstance** out_InstanceInterface ); ERuntimeErrorCode RegisterInstance(     WCHAR* in_InstanceName,     IInstance** out_InstanceInterface ); ERuntimeErrorCode RegisterInstance(     ECPUType in_CPUType,     IInstance** out_InstanceInterface ); ERuntimeErrorCode RegisterInstance(     ECPUType in_CPUType,     WCHAR* in_InstanceName,     IInstance** out_InstanceInterface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• ECPUType in_CPUType:                     <p>Defines which CPU type is simulated at the start of the instance. The default setting is "SRCT_1500_Unspecified".</p> <p>When a different CPU type is loaded via STEP 7 or from the Virtual Memory Card, this CPU type applies.</p> </li> <li>• WCHAR* in_InstanceName:                     <p>Name to be assigned to the instance. Every instance must have a unique name. If no name is assigned when registering a new instance, the instance is given the name "Instance_#" (# is the ID of the instance). If this name already exists, the name "Instance_#. #" is used, in which the second # is a counter that is incremented until the name is unique. The length of the name must be less than DINSTANCE_NAME_LENGTH. See Data types (Page 271).</p> </li> <li>• IInstance** out_InstanceInterface:                     <p>Pointer to a Simulation Runtime interface pointer. The pointer <b>must</b> be initialized with NULL. The interface is created within the function.</p> </li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The interface is disconnected from the remote Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_WRONG_ARGUMENT	The name or the IInstance pointer is invalid.
	SREC_LIMIT_REACHED	There are already 16 instances registered in Runtime Manager.
	SREC_ALREADY_EXISTS	An instance with this name already exists.



Example C++	<pre> ISimulationRuntimeManager * api = NULL; ERuntimeErrorCode result = Initialize(&amp;api);  // Example: How To Create And Register An Instance // And To Get An Interface Of The Instance The Same Time IInstance* psa = NULL; if (result == SREC_OK) {     result = api-&gt;RegisterInstance(&amp;psa); } </pre>
-------------	---

**Note****Native C++**

If you no longer require the interface, delete it.

See DestroyInterface() (Page 82).

Table 7- 288 RegisterInstance() - .NET (C#)

Syntax	<pre> IInstance RegisterInstance(); IInstance RegisterInstance(     string in_InstanceName ); IInstance RegisterInstance(     ECPUType in_CPUType ); IInstance RegisterInstance(     ECPUType in_CPUType     string in_InstanceName ); </pre>
Parameters	<ul style="list-style-type: none"> <li>• ECPUType in_CPUType:                  Defines which CPU type is simulated at the start of the instance. The default setting is "ECPUType.Unspecified".                  When a different CPU type is loaded via STEP 7 or from the Virtual Memory Card, this CPU type applies.</li> <li>• string in_InstanceName:                  Name to be assigned to the instance. Every instance must have a unique name. If no name is assigned when registering a new instance, the instance is given the name "Instance_#" (# is the ID of the instance). If this name already exists, the name "Instance_#. #" is used, in which the second # is a counter that is incremented until the name is unique. The length of the name must be less than DINSTANCE_NAME_LENGTH. See Data types (Page 271).</li> </ul>
Return values	If the function is successful, an interface of a virtual controller. Otherwise, a Null pointer.

Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The interface is disconnected from the remote Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.WrongArgument	The name is invalid.
	ERuntimeErrorCode.LimitReached	There are already 16 instances registered in Runtime Manager.
ERuntimeErrorCode.AlreadyExists	An instance with this name already exists.	

### RegisterCustomInstance()

Registers a new instance of a virtual controller in Runtime Manager. Creates and returns an interface of this instance.

Table 7- 289 RegisterCustomInstance() - Native C++

Syntax	<pre>ERuntimeErrorCode RegisterCustomInstance(     WCHAR* in_VplcDll,     IInstance** out_InstanceInterface ); ERuntimeErrorCode RegisterCustomInstance(     WCHAR* in_VplcDll,     WCHAR* in_InstanceName,     IInstance** out_InstanceInterface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>WCHAR* in_VplcDll: The complete path to the DLL of the virtual controller that Siemens.Simatic.Simulation.Runtime.Instance.exe loads at PowerOn.</li> <li>WCHAR* in_InstanceName: Name to be assigned to the instance. Every instance must have a unique name. If no name is assigned when registering a new instance, the instance is given the name "Instance_#" (# is the ID of the instance). If this name already exists, the name "Instance_#. #" is used, in which the second # is a counter that is incremented until the name is unique. The length of the name must be less than DINSTANCE_NAME_LENGTH. See Data types (Page 271).</li> <li>IInstance** out_InstanceInterface: Pointer to a Simulation Runtime interface pointer. The pointer must be initialized with NULL. The interface is created within the function.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The interface is disconnected from the remote Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_WRONG_ARGUMENT	The DLL name, the instance name or the IInstance pointer is invalid.
	SREC_LIMIT_REACHED	There are already 16 instances registered in Runtime Manager.
SREC_ALREADY_EXISTS	An instance with this name already exists.	

Example C++	<pre> ISimulationRuntimeManager * api = NULL; ERuntimeErrorCode result = Initialize(&amp;api);  // Example: How To Create And Register An Instance // And To Get An Interface Of The Instance The Same Time IInstance* psa = NULL; if (result == SREC_OK) {     result = api-&gt;RegisterCustomInstance("C:\\Temp\\vplc.dll"); } </pre>
-------------	---

**Note****Native C++**

If you no longer require the interface, delete it.

See DestroyInterface() (Page 82).

Table 7- 290 RegisterCustomInstance() - .NET (C#)

Syntax	<pre> IInstance RegisterCustomInstance(     string in_VplcDll ); IInstance RegisterCustomInstance(     string in_VplcDll,     string in_InstanceName ); </pre>	
Parameters	<ul style="list-style-type: none"> <li>• string in_VplcDll: The complete path to the DLL of the virtual controller that Siemens.Simatic.Simulation.Runtime.Instance.exe loads at PowerOn.</li> <li>• string in_InstanceName: Name to be assigned to the instance. Every instance must have a unique name. If no name is assigned when registering a new instance, the instance is given the name "Instance_#" (# is the ID of the instance). If this name already exists, the name "Instance_#. #" is used, in which the second # is a counter that is incremented until the name is unique. The length of the name must be less than DINSTANCE_NAME_LENGTH. See Data types (Page 271).</li> </ul>	
Return values	If the function is successful, an interface of a virtual controller; otherwise a Null pointer.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	<b>Runtime error code</b>	<b>Condition</b>
	ERuntimeError-Code.InterfaceRemoved	The interface is disconnected from the remote Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.WrongArgument	The name or the ID is invalid.
	ERuntimeErrorCode.LimitReached	There are already 16 instances registered in Runtime Manager.
	ERuntimeErrorCode.AlreadyExists	An instance with this name already exists.

**CreateInterface()**

Creates and returns an interface of an already registered instance of a virtual controller.

The instance could have been registered via the application or another application that uses the Simulation Runtime API.

Table 7- 291 CreateInterface() - Native C++

Syntax	<pre>ERuntimeErrorCode CreateInterface(     WCHAR* in_InstanceName,     IInstance** out_InstanceInterface ); ERuntimeErrorCode CreateInterface(     INT32 in_InstanceID,     IInstance** out_InstanceInterface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• INT32 in_InstanceID: The ID of the registered instance from which you want to receive the interface.</li> <li>• WCHAR* in_InstanceName: The name of the registered instance from which you want to receive the interface.</li> <li>• IInstance** out_InstanceInterface: Pointer to a Simulation Runtime interface pointer. The pointer must be initialized with NULL. The interface is created within the function.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_INTERFACE_REMOVED	The interface is disconnected from the remote Runtime Manager.
	SREC_TIMEOUT	The function does not return on time.
	SREC_WRONG_ARGUMENT	The name, the ID or the IInstance- pointer is invalid.
SREC_DOES_NOT_EXIST	The instance is not registered in Runtime Manager.	
Example C++	<pre>ISimulationRuntimeManager * api = NULL; ERuntimeErrorCode result = Initialize(&amp;api);  IInstance* psal = NULL; IInstance* psa2 = NULL; if (result == SREC_OK) {     result = api-&gt;CreateInterface(0, &amp;psal);     result = api-&gt;CreateInterface(0, &amp;psa2); // psa2 will be the same as psal } }</pre>	
Example C++	<pre>ISimulationRuntimeManager * api = NULL; ERuntimeErrorCode result = Initialize(&amp;api);  IInstance* psa = NULL; if (result == SREC_OK) {     result = api-&gt;CreateInterface(L"My SimulationRuntime Instance",     &amp;psa); } }</pre>	

**Note****Native C++**

If you no longer require the interface, delete it.

See DestroyInterface() (Page 82)

Table 7- 292 CreateInterface() - .NET (C#)

Syntax	<pre>IInstance CreateInterface(     string in_InstanceName ); IInstance CreateInterface(     INT32 in_InstanceID );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• INT32 in_InstanceID: The ID of the registered instance from which you want to receive the interface.</li> <li>• string in_InstanceName: The name of the registered instance from which you want to receive the interface.</li> </ul>	
Return values	If the function is successful, an interface of a virtual controller; otherwise a Null pointer.	
Exceptions	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime error code	Condition
	ERuntimeError-Code.InterfaceRemoved	The interface is disconnected from the remote Runtime Manager.
	ERuntimeErrorCode.Timeout	The function does not return on time.
	ERuntimeErrorCode.WrongArgument	The name or the ID is invalid.
	ERuntimeErrorCode.DoesNotExists	The instance is not registered in Runtime Manager.

### 7.7.3 Events

#### 7.7.3.1 OnConnectionLost

##### Description

The event is triggered when the connection to the Remote Runtime Manager has been terminated.

##### OnConnectionLost

Registers or unregisters an event handler method.

Table 7- 293 OnConnectionLost - .NET (C#)

Syntax	<code>event Delegate_IRRTM OnConnectionLost;</code>
Parameters	None. See Delegate_IRRTM (Page 269)
Return values	None
Exceptions	None
Note	The event handler method runs in a separate thread.

##### RegisterOnConnectionLostCallback()

When the event occurs, the registered callback function is called. Only one callback function can be registered for the event. Registering a new callback function causes the previous callback function to be unregistered.

Table 7- 294 RegisterOnConnectionLostCallback() - Native C++

Syntax	<code>void RegisterOnConnectionLostCallback( EventCallback_IRRTM in_CallbackFunction );</code>
Parameters	<ul style="list-style-type: none"> <li>EventCallback_IRRTM in_CallbackFunction: A callback function that subscribes to an event. See EventCallback_IRRTM (Page 263).</li> </ul>
Return values	None
Note	The callback function runs in a separate thread.

## RegisterOnConnectionLostEvent()

When the event occurs, the registered event object is set to the signaled state. Only one event object can be registered for the event. Registration of a new event object causes the previous event object to be deleted.

Table 7- 295 RegisterOnConnectionLostEvent() - Native C++

Syntax	<pre>void RegisterOnConnectionLostEvent(); void RegisterOnConnectionLostEvent( HANDLE* in_Event );</pre>
Parameters	<ul style="list-style-type: none"> <li>None: An internal event object is registered.</li> <li>HANDLE* in_Event: A handle for a user-specific event object. The event object is registered.</li> </ul>
Return values	None

Table 7- 296 RegisterOnConnectionLostEvent() - .NET (C#)

Syntax	<pre>void RegisterOnConnectionLostEvent();</pre>
Parameters	None
Return values	None

## UnregisterOnConnectionLostCallback()

Unregisters the callback function. When the event occurs, no callback function is called.

Table 7- 297 UnregisterOnConnectionLostCallback() - Native C++

Syntax	<pre>void UnregisterOnConnectionLostCallback();</pre>
Parameters	None
Return values	None

### UnregisterOnConnectionLostEvent()

Unregisters the event object.

Table 7- 298 UnregisterOnConnectionLostEvent() - Native C++

Syntax	<code>void UnregisterOnConnectionLostEvent();</code>
Parameters	None
Return values	None

Table 7- 299 UnregisterOnConnectionLostEvent() - .NET (C#)

Syntax	<code>void UnregisterOnConnectionLostEvent();</code>
Parameters	None
Return values	None

### WaitForOnConnectionLostEvent()

The function blocks the program until the registered event object is in the signaled state or the timeout interval is exceeded.

Table 7- 300 WaitForOnConnectionLostEvent() - Native C++

Syntax	<code>bool WaitForOnConnectionLostEvent(); bool WaitForOnConnectionLostEvent(   UINT32 in_Time_ms ) ;</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UINT32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>

Table 7- 301 WaitForOnConnectionLostEvent() - .NET (C#)

Syntax	<code>bool WaitForOnConnectionLostEvent(); bool WaitForOnConnectionLostEvent(   UInt32 in_Time_ms ) ;</code>
Parameters	<ul style="list-style-type: none"> <li>None: The time limit is set to INFINITE.</li> <li>UInt32 in_Time_ms: Value for the time limit in milliseconds.</li> </ul>
Return values	<ul style="list-style-type: none"> <li>true: If the event object was set to the signaled state.</li> <li>false: If no event was received during the defined time limit.</li> </ul>



## 7.8 Data types

---

### Note

#### Unsupported data types

The Runtime API does not support the `STRING` and `WSTRING` data types.

---

### Supported data types

In S7-PLCSIM Advanced V1.0, the Runtime API supports the data types of the S7-1500 CPUs.

### Converting data types

When writing, data types are not transferred BCD-coded but mapped onto primitive data types.

The data types Counter, Date and Time must be transferred to the API BCD-coded so that the values are written to the counter and no incorrect values are returned when reading.

For these data types, you must perform a BCD conversion before writing and a BCD back-conversion after reading.

#### Example:

If the value 999 is transferred to the API as 2457<sub>H</sub>, then `Write` modifies the value 2457<sub>H</sub> to 999. Without BCD conversion, there is no `UInt16` value and `Write` writes no value at all.

### Additional information

For information on data types and conversion, refer to section "Data types" in the STEP 7 V14 System Manual

(<https://support.industry.siemens.com/cs/document/109011420/step-7-professional-v13-1?dti=0&lc=en-WW // XmlEditor.InternalXmlClipboard:d59d29c4-ec28-6b28-3f63-754735fb2e2a>).

## 7.8.1 DLL import functions (Native C++)

### 7.8.1.1 ApiEntry\_Initialize

#### Description

Type of the central entry point for the API DLL.

Table 7- 302 ApiEntry\_Initialize - Native C++

Syntax	<pre>typedef HRESULT (*ApiEntry_Initialize) (     ISimulationRuntimeManager** out_RuntimeManagerInterface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>ISimulationRuntimeManager** out_SimulationRuntimeManagerInterface: Pointer to a Runtime Manager interface pointer. The pointer must be initialized with ZERO. The interface is created within the function.</li> <li>UINT32 in_InterfaceVersion: Version of the API interface to be downloaded: API_DLL_INTERFACE_VERSION.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_WRONG_ARGUMENT	The pointer to the Runtime Manager interface is ZERO.
	SREC_WRONG_VERSION	The version of the interface in use does not match the version of the API DLL.
	SREC_CONNECTION_ERROR	Unable to establish a connection to the Runtime Manager.

### 7.8.1.2 ApiEntry\_DestroyInterface

#### Description

Type of the entry point for DestroyInterface (Page 82).

Table 7- 303 ApiEntry\_DestroyInterface - Native C++

Syntax	<pre>typedef HRESULT (*ApiEntry_DestroyInterface) (     IBaseInterface* in_Interface );</pre>	
Parameters	<ul style="list-style-type: none"> <li>IBaseInterface* in_Interface: The interface to be deleted.</li> </ul>	
Return values	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_WRONG_ARGUMENT	The pointer to the interface is ZERO.

## 7.8.2 Event callback functions (Native C++)

### 7.8.2.1 EventCallback\_VOID

#### Description

Table 7- 304 EventCallback\_VOID - Native C++

Syntax	<code>typedef void (*EventCallback_VOID) ();</code>
Parameters	None
Return values	None

### 7.8.2.2 EventCallback\_II\_SREC\_ST

#### Description

Table 7- 305 EventCallback\_II\_SREC\_ST - Native C++

Syntax	<code>typedef void (*EventCallback_II_SREC_ST) (   IInstance* in_Sender,   ERuntimeErrorCode in_ErrorCode,   SYSTEMTIME in_SystemTime );</code>
Parameters	<ul style="list-style-type: none"> <li>• IInstance* in_Sender: An interface of the instance that receives this event.</li> <li>• ERuntimeErrorCode in_ErrorCode: A possible error code.</li> <li>• SYSTEMTIME in_SystemTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> </ul>
Return values	None

### 7.8.2.3 EventCallback\_II\_SREC\_ST\_SROS\_SROS

#### Description

Table 7- 306 EventCallback\_II\_SREC\_ST\_SROS\_SROS - Native C++

Syntax	<pre>typedef void (*EventCallback_II_SREC_ST_SROS_SROS) (     IInstance* in_Sender,     ERuntimeErrorCode in_ErrorCode,     SYSTEMTIME in_SystemTime,     EOperatingState in_PrevState,     EOperatingState in_OperatingState );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• IInstance* in_Sender: An interface of the instance that receives this event.</li> <li>• ERuntimeErrorCode in_ErrorCode: A possible error code.</li> <li>• SYSTEMTIME in_SystemTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> <li>• EOperatingState in_PrevState: The operating state before the change.</li> <li>• EOperatingState in_OperatingState: The current operating state.</li> </ul>	
Return values	None	
Error codes	Runtime error code	Condition
	SREC_OK	The function is successful.
	SREC_WARNING_TRIAL_MODE_ACTIVE	No license available. You can use the instance without restrictions for a period of one hour. Afterwards, the instance is shut down.
	SREC_LICENSE_NOT_FOUND	Test mode has expired.
	SREC_COMMUNICATION_INTERFACE_NOT_AVAILABLE	A problem has occurred with the selected communication interface. Check your settings.

## 7.8.2.4 EventCallback\_II\_SREC\_ST\_SRLT\_SRLM

## Description

Table 7- 307 EventCallback\_II\_SREC\_ST\_SRLT\_SRLM - Native C++

Syntax	<pre>typedef void (*EventCallback_II_SREC_ST_SRLT_SRLM) (     IInstance* in_Sender,     ERuntimeErrorCode in_ErrorCode,     SYSTEMTIME in_SystemTime,     ELEDType in_LEDType,     ELEDMode in_LEDMode, );</pre>
Parameters	<ul style="list-style-type: none"> <li>• IInstance* in_Sender: An interface of the instance that receives this event.</li> <li>• ERuntimeErrorCode in_ErrorCode: A possible error code.</li> <li>• SYSTEMTIME in_SystemTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> <li>• ELEDType in_LEDType: The LED type that changed its state.</li> <li>• ELEDMode in_LEDMode: The new state of the LED display.</li> </ul>
Return values	None

7.8.2.5 EventCallback\_II\_SREC\_ST\_INT64\_UINT32

Description

Table 7- 308 EventCallback\_II\_SREC\_ST\_INT64\_UINT32 - Native C++

Syntax	<pre>typedef void (*EventCallback_II_SREC_ST_INT64_UINT32) (     IInstance* in_Sender,     ERuntimeErrorCode in_ErrorCode,     SYSTEMTIME in_SystemTime,     INT64 in_CycleTime_ns,     UINT32 in_CycleCount, );</pre>
Parameters	<ul style="list-style-type: none"> <li>• IInstance* in_Sender: An interface of the instance that receives this event.</li> <li>• ERuntimeErrorCode in_ErrorCode: A possible error code.</li> <li>• SYSTEMTIME in_SystemTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> <li>• INT64 in_CycleTime_ns: The virtual time (in nanoseconds) since the last cycle control point. If the events are triggered faster than they are received, multiple events are combined into one event. In this case, the value contains the virtual time (in nanoseconds) since the last event was received.</li> <li>• UINT32 in_CycleCount: The number of cycles since the last cycle control point. If the events are triggered faster than they are received, multiple events are combined into one event. In this case, this value contains the number of cycles since the last event was received.</li> </ul>
Return values	None

### 7.8.2.6 EventCallback\_IRRTM

#### Description

Table 7- 309 EventCallback\_IRRTM - Native C++

Syntax	<pre>typedef void (*EventCallback_IRRTM) (     IRemoteRuntimeManager* in_Sender );</pre>
Parameters	<ul style="list-style-type: none"> <li>IRemoteRuntimeManager* in_Sender:</li> </ul> <p>An interface of the remote Runtime Manager that receives this event.</p>
Return values	None

### 7.8.2.7 EventCallback\_SRCC\_UINT32\_UINT32\_INT32

#### Description

Table 7- 310 EventCallback\_SRCC\_UINT32\_UINT32\_INT32 - Native C++

Syntax	<pre>ERuntimeConfigChanged in_RuntimeConfigChanged, UINT32 in_Param1, UINT32 in_Param2, INT32 in_Param3 );</pre>			
Parameters	ERuntimeConfigChanged in_RuntimeConfigChanged	UInt32 in_Param1	UInt32 in_Param2	Int32 in_Param3
	SRCC_INSTANCE_REGISTERED	-	-	ID of the registered instance
	SRCC_INSTANCE_UNREGISTERED	-	-	ID of the unregistered instance
	SRCC_CONNECTION_OPENED	IP of the remote Runtime Manager	Port of the remote Runtime Manager	-
	SRCC_CONNECTION_CLOSED	IP of the remote Runtime Manager	Port of the remote Runtime Manager	-
	SRCC_PORT_OPENED	The open port	-	-
	SRCC_PORT_CLOSED	-	-	-
Return values	None			

7.8.2.8 EventCallback\_II\_SREC\_ST\_SRICC\_UINT32\_UINT32\_UINT32\_UINT32

Description

Table 7- 311 EventCallback\_II\_SREC\_ST\_SRICC\_UINT32\_UINT32\_UINT32\_UINT32 - Native C++

Syntax	<pre>typedef void (*EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32) (   IInstance* in_Sender,   ERuntimeErrorCode in_ErrorCode, SYSTEMTIME in_SystemTime,   EInstanceConfigChanged in_InstanceConfigChanged,   UINT32 in_Param1,   UINT32 in_Param2,   UINT32 in_Param3,   UINT32 in_Param4 );</pre>				
Parameters	<ul style="list-style-type: none"> <li>IInstance in_Sender: An interface of the instance that receives this event.</li> <li>ERuntimeErrorCode in_ErrorCode: A possible error code.</li> <li>SYSTEMTIME in_SystemTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> </ul>				
	EInstanceConfigChanged in_InstanceConfigChanged	UINT32 in_Param1	UINT32 in_Param2	UINT32 in_Param3	UINT32 in_Param4
	SRICC_HARDWARE_SOFTWARE_CHANGED	-	-	-	-
	SRICC_IP_CHANGED	The ID of the interface	The new IP	The new subnet mask	The new standard gateway
Return values	None				



## 7.8.3 Delegate definitions (managed code)

### 7.8.3.1 Delegate\_Void

#### Description

Table 7- 312 Delegate\_Void - .NET (C#)

Syntax	<code>delegate void Delegate_Void();</code>
Parameters	None
Return values	None

### 7.8.3.2 Delegate\_II\_EREC\_DT

#### Description

Table 7- 313 Delegate\_II\_EREC\_DT - .NET (C#)

Syntax	<code>delegate void Delegate_II_EREC_DT (   IInstance in_Sender,   ERuntimeErrorFCode in_ErrorCode,   DateTime in_DateTime );</code>
Parameters	<ul style="list-style-type: none"> <li>• <code>IInstance in_Sender:</code> An interface of the instance that receives this event.</li> <li>• <code>ERuntimeErrorFCode in_ErrorCode:</code> A possible error code.</li> <li>• <code>DateTime in_DateTime:</code> The virtual system time of the virtual controller at the time when this event was triggered.</li> </ul>
Return values	None

### 7.8.3.3 Delegate\_II\_EREC\_DT\_EOS\_EOS

#### Description

Table 7- 314 Delegate\_II\_EREC\_DT\_EOS\_EOS - .NET (C#)

Syntax	<pre>delegate void Delegate_II_EREC_DT_EOS_EOS (     IInstance in_Sender,     ERuntimeErrorCode in_ErrorCode,     DateTime in_DateTime,     EOperatingState in_PrevState,     EOperatingState in_OperatingState );</pre>	
Parameters	<ul style="list-style-type: none"> <li>• IInstance in_Sender: An interface of the instance that receives this event.</li> <li>• ERuntimeErrorCode in_ErrorCode: A possible error code.</li> <li>• DateTime in_DateTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> <li>• EOperatingState in_PrevState: The operating state before the change.</li> <li>• EOperatingState in_OperatingState: The current operating state.</li> </ul>	
Return values	None	
Error codes	Runtime error code	Condition
	ERuntimeErrorCode.OK	The function is successful.
	ERuntimeError-Code.WarningTrialModeActive	No license available. You can use the instance without restrictions for a period of one hour. Afterwards, the instance is shut down.
	ERuntimeError-Code.LicenseNotFound	Test mode has expired.
	ERuntimeError-Code.CommunicationInterfaceNotAvailable	A problem has occurred with the selected communication interface. Check your settings.

### 7.8.3.4 Delegate\_II\_EREC\_DT\_ELT\_ELM

#### Description

Table 7- 315 Delegate\_II\_EREC\_DT\_ELT\_ELM - .NET (C#)

Syntax	<pre>delegate void Delegate_II_EREC_DT_ELT_ELM(     IInstance in_Sender,     ERuntimeErrorcode in_ErrorCode,     DateTime in_DateTime,     ELEDType in_LEDType,     ELEDMode in_LEDMode, );</pre>
Parameters	<ul style="list-style-type: none"> <li>• IInstance in_Sender: An interface of the instance that receives this event.</li> <li>• ERuntimeErrorcode in_ErrorCode: A possible error code.</li> <li>• DateTime in_DateTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> <li>• ELEDType in_LEDType: The LED type that changed its state.</li> <li>• ELEDMode in_LEDMode: The new state of the LED display.</li> </ul>
Return values	None

### 7.8.3.5 Delegate\_II\_EREC\_DT\_INT64\_UINT32

#### Description

Table 7- 316 Delegate\_II\_EREC\_DT\_INT64\_UINT32 - .NET (C#)

Syntax	<pre>delegate void Delegate_II_EREC_DT_INT64_UINT32 (     IInstance in_Sender,     ERuntimeErrorCode in_ErrorCode,     DateTime in_DateTime,     Int64 in_CycleTime_ns,     UInt32 in_CycleCount, );</pre>
Parameters	<ul style="list-style-type: none"> <li>• IInstance in_Sender: An interface of the instance that receives this event.</li> <li>• ERuntimeErrorCode in_ErrorCode: A possible error code.</li> <li>• DateTime in_DateTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> <li>• Int64 in_CycleTime_ns: The virtual time (in nanoseconds) since the last cycle control point. If the events are triggered faster than they are received, multiple events are combined into one event. In this case, the value contains the virtual time (in nanoseconds) since the last event was received.</li> <li>• UInt32 in_CycleCount: The number of cycles since the last cycle control point. If the events are triggered faster than they are received, multiple events are combined into one event. In this case, this value contains the number of cycles since the last event was received.</li> </ul>
Return values	None

### 7.8.3.6 Delegate\_IRRTM

#### Description

Table 7- 317 Delegate\_IRRTM - .NET (C#)

Syntax	<pre>delegate void Delegate_IRRTM(     IRemoteRuntimeManager in_Sender, );</pre>
Parameters	<ul style="list-style-type: none"> <li>IRemoteRuntimeManager in_Sender: An interface of the remote Runtime Manager that receives this event.</li> </ul>
Return values	None

### 7.8.3.7 Delegate\_SRCC\_UINT32\_UINT32\_INT32

#### Description

Table 7- 318 Delegate\_SRCC\_UINT32\_UINT32\_INT32 - .NET (C#)

Syntax	<pre>delegate void Delegate_SRCC_UINT32_UINT32_INT32(     ERuntimeConfigChanged in_RuntimeConfigChanged,     UInt32 in_Param1,     UInt32 in_Param2,     Int32 in_Param3 );</pre>			
Parameters	ERuntimeConfigChanged in_RuntimeConfigChanged	UInt32 in_Param1	UInt32 in_Param2	Int32 in_Param3
	InstanceRegistered	-	-	ID of the registered instance
	InstanceUnregistered	-	-	ID of the unregistered instance
	ConnectionOpened	IP of the Remote Runtime Manager	Port of the remote Runtime Manager	-
	ConnectionClosed	IP of the Remote Runtime Manager	Port of the remote Runtime Manager	-
	PortOpened	The open port	-	-
	PortClosed	-	-	-
Return values	None			

7.8.3.8 Delegate\_II\_EREC\_DT\_SRICC\_UINT32\_UINT32\_UINT32\_UINT32

Description

Table 7- 319 Delegate\_II\_EREC\_DT\_SRICC\_UINT32\_UINT32\_UINT32\_UINT32 - .NET (C#)

Syntax	<pre>delegate void Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32(     IInstance in_Sender,     ERuntimeErrorCode in_ErrorCode,     DateTime in_DateTime,     EInstanceConfigChanged in_InstanceConfigChanged,     UInt32 in_Param1,     UInt32 in_Param2,     UInt32 in_Param3,     UInt32 in_Param4 );</pre>				
Parameters	<ul style="list-style-type: none"> <li>IInstance in_Sender: An interface of the instance that receives this event.</li> <li>ERuntimeErrorCode in_ErrorCode: A possible error code.</li> <li>DateTime in_DateTime: The virtual system time of the virtual controller at the time when this event was triggered.</li> </ul>				
	EInstanceConfigChanged in_InstanceConfigChanged	UInt32 in_Param1	UInt32 in_Param2	UInt32 in_Param3	UInt32 in_Param4
	HardwareSoftwareChanged	-	-	-	-
	IPChanged	The ID of the interface	The new IP	The new subnet mask	The new standard gateway
Return values	None				

## 7.8.4 Definitions and constants

The following identifiers are used in the API:

Table 7- 320 Definitions and constants - Native C++

Identifier	Value	Description
DINSTANCE_NAME_MAX_LENGTH	64	The unique name of an instance must be less than this value.
DSTORAGE_PATH_MAX_LENGTH	130	The maximum path length to the virtual memory card. Including ZERO termination.
DTAG_NAME_MAX_LENGTH	300	The maximum length of the name of a PLC tag. Including ZERO termination.
DTAG_ARRAY_DIMENSION	6	The maximum number of dimension for a multi-dimensional field.
DCONTROLLER_NAME_MAX_LENGTH	128	The maximum length of the controller name. Including ZERO termination.
DCONTROLLER_SHORT_DESIGNATION_MAX_LENGTH	32	The maximum length of the abbreviation of the controller (CPU type). Including ZERO termination.

Table 7- 321 Definitions and constants - .NET (C#)

Identifier	Value	Description
RuntimeConstants.InstanceNameLength	64	The unique name of an instance must be less than this value.
RuntimeConstants.StoragePathMaxLength	130	The maximum path length to the virtual memory card. Including ZERO termination.
RuntimeConstants.TagNameMaxLength	300	The maximum length of the name of a PLC tag. Including ZERO termination.
RuntimeConstants.TagArrayDimension	6	The maximum number of dimension for a multi-dimensional field.
RuntimeConstants.ControllerNameMaxLength	128	The maximum length of the controller name. Including ZERO termination.
RuntimeConstants.ControllerShortDesignationMaxLength	32	The maximum length of the abbreviation of the controller (CPU type). Including ZERO termination.

## 7.8.5 Unions (Native C++)

### 7.8.5.1 UIP

#### Description

Contains an IPv4 address.

Table 7- 322 UIP - Native C++

Syntax	<pre>union UIP {     DWORD IP;     BYTE IPs[4]; };</pre>
Member	<ul style="list-style-type: none"> <li>• <b>DWORD IP:</b> The IP address in a single DWORD</li> <li>• <b>BYTE IPs[4]:</b> The four elements of IP in descending order</li> </ul>
Example	<p>Example for an IP address: 192.168.0.1</p> <p>UIP.IP = 0xC0A80001</p> <p>UIP.IPs[3] = 192, UIP.IPs[2] = 168, UIP.IPs[1] = 0, UIP.IPs[0] = 1</p>

### 7.8.5.2 UDataValue

#### Description

Contains the value of a PLC tag.

Table 7- 323 UDataValue - Native C++

Syntax	<pre>union UDataValue {     bool Bool;     INT8 Int8;     INT16 Int16;     INT32 Int32;     INT64 Int64;     UINT8 UInt8;     UINT16 UInt16;     UINT32 UInt32;     UINT64 UInt64;     float Float;     double Double;     CHAR Char;     WCHAR WChar; };</pre>
--------	---



Member	
	<ul style="list-style-type: none"> <li data-bbox="609 251 869 325">• <code>bool Bool:</code> 1 byte boolean value</li> <li data-bbox="609 342 895 417">• <code>INT8 Int8:</code> 1 byte integer with sign</li> <li data-bbox="609 434 895 508">• <code>INT16 Int16:</code> 2 byte integer with sign</li> <li data-bbox="609 525 895 600">• <code>INT32 Int32:</code> 4 byte integer with sign</li> <li data-bbox="609 617 895 691">• <code>INT64 Int64:</code> 8 byte integer with sign</li> <li data-bbox="609 708 927 783">• <code>UINT8 UInt8:</code> 1 byte integer without sign</li> <li data-bbox="609 800 927 874">• <code>UINT16 UInt16:</code> 2 byte integer without sign</li> <li data-bbox="609 891 927 966">• <code>UINT32 UInt32:</code> 4 byte integer without sign</li> <li data-bbox="609 983 927 1057">• <code>UINT64 UInt64:</code> 8 byte integer without sign</li> <li data-bbox="609 1074 922 1149">• <code>float Float:</code> 4 byte floating-point value</li> <li data-bbox="609 1166 922 1240">• <code>double Double:</code> 8 byte floating-point value</li> <li data-bbox="609 1257 885 1332">• <code>CHAR Char:</code> 1 byte value character</li> <li data-bbox="609 1349 885 1423">• <code>WCHAR WChar:</code> 2 byte value character</li> </ul>

## 7.8.6 Structures

### 7.8.6.1 SDataValue

#### Description

The structure contains the value and type of a PLC tag.

Table 7- 324 SDataValue - Native C++

Syntax	<pre>struct SDataValue {     UDataValue Value;     EDataType Type; };</pre>
Member	<ul style="list-style-type: none"> <li>UDataValue Value: The value of the PLC tags</li> <li>EPrimitiveDataType Type: Type of PLC tag</li> </ul>

Table 7- 325 SDataValue - .NET (C#)

Syntax	<pre>struct SDataValue {     bool Bool { get; set; }     Int8 Int8 { get; set; }     Int16 Int16 { get; set; }     Int32 Int32 { get; set; }     Int64 Int64 { get; set; }     UInt8 UInt8 { get; set; }     UInt16 UInt16 { get; set; }     UInt32 UInt32 { get; set; }     UInt64 UInt64 { get; set; }     float Float { get; set; }     double Double { get; set; }     sbyte Char { get; set; }     char WChar { get; set; }     EPrimitiveDataType Type { get; set; } }</pre>
--------	--

Member	
	<ul style="list-style-type: none"> <li data-bbox="609 244 869 308">• <code>bool Bool:</code> 1 byte boolean value</li> <li data-bbox="609 319 893 383">• <code>Int8 Int8:</code> 1 byte integer with sign</li> <li data-bbox="609 393 893 457">• <code>Int16 Int16:</code> 2 byte integer with sign</li> <li data-bbox="609 468 893 532">• <code>Int32 Int32:</code> 4 byte integer with sign</li> <li data-bbox="609 542 893 606">• <code>Int64 Int64:</code> 8 byte integer with sign</li> <li data-bbox="609 617 925 680">• <code>UInt8 UInt8:</code> 1 byte integer without sign</li> <li data-bbox="609 691 925 755">• <code>UInt16 UInt16:</code> 2 byte integer without sign</li> <li data-bbox="609 766 925 829">• <code>UInt32 UInt32:</code> 4 byte integer without sign</li> <li data-bbox="609 840 925 904">• <code>UInt64 UInt64:</code> 8 byte integer without sign</li> <li data-bbox="609 915 925 978">• <code>float Float:</code> 4 byte floating-point value</li> <li data-bbox="609 989 925 1053">• <code>double Double:</code> 8 byte floating-point value</li> <li data-bbox="609 1064 885 1127">• <code>sbyte Char:</code> 1 byte value character</li> <li data-bbox="609 1138 885 1202">• <code>char WChar:</code> 2 byte value character</li> <li data-bbox="609 1212 949 1276">• <code>EPrimitiveDataType Type:</code> Type of PLC tag</li> </ul>

### 7.8.6.2 SDataValueByAddress

#### Description

This structure represents a PLC tag that can be accessed via its address.

Table 7- 326 SDataValueByAddress - Native C++

Syntax	<pre>struct SDataValueByAddress {     UINT32 Offset;     UINT8 Bit;     SDataValue DataValue; };</pre>
--------	--

Table 7- 327 SDataValueByAddress - .NET (C#)

Syntax	<pre>struct SDataValueByAddress {     UInt32 Offset;     UInt8 Bit;     SDataValue DataValue; }</pre>
--------	---

### 7.8.6.3 SDataValueByName

#### Description

This structure represents a PLC tag that can be called by name.

Table 7- 328 SDataValueByName - Native C++

Syntax	<pre>struct SDataValueByName {     WCHAR Name[DTAG_NAME_MAX_LENGTH];     SDataValue DataValue; };</pre>
--------	---

Table 7- 329 SDataValueByName - .NET (C#)

Syntax	<pre>struct SDataValueByName {     String Name;     SDataValue DataValue; }</pre>
--------	---

#### 7.8.6.4 SConnectionInfo

##### Description

This structure contains the IP address and port of a TCP/IP connection.

Table 7- 330 SConnectionInfo - Native C++

Syntax	<pre>struct SConnectionInfo {     UIP IP;     WORD Port; };</pre>
--------	---

Table 7- 331 SConnectionInfo - .NET (C#)

Syntax	<pre>struct SConnectionInfo {     SIP IP;     UInt16 Port; }</pre>
--------	--

### 7.8.6.5 SInstanceInfo

#### Description

This structure contains an IPv4 address.

Table 7- 332 SInstanceInfo - Native C++

Syntax	<pre>struct SInstanceInfo {     INT32 ID;     WCHAR Name[DINSTANCE_NAME_MAX_LENGTH]; };</pre>
Member	<ul style="list-style-type: none"> <li>• INT32 ID: The ID of the instance</li> <li>• WCHAR Name[DINSTANCE_NAME_MAX_LENGTH]: The name of the instance</li> </ul>

Table 7- 333 SInstanceInfo - .NET (C#)

Syntax	<pre>struct SInstanceInfo {     Int32 ID;     String Name; }</pre>
Member	<ul style="list-style-type: none"> <li>• Int32 ID: The ID of the instance</li> <li>• String name: The name of the instance</li> </ul>

### 7.8.6.6 SDimension

#### Description

This structure contains information about the dimension of a field.

Table 7- 334 SDimension - Native C++

Syntax	<pre>struct SDimension {   INT32 StartIndex;   UINT32 Count; };</pre>
--------	---

Table 7- 335 SDimension - .NET (C#)

Syntax	<pre>struct SDimension {   Int32 StartIndex;   UInt32 Count; }</pre>
--------	--

### 7.8.6.7 STagInfo

#### Description

This structure contains information about a PLC tag.

Table 7- 336 STagInfo - Native C++

Syntax	<pre> struct STagInfo {     WCHAR Name[DTAG_NAME_MAX_LENGTH];     EArea Area;     EDataType DataType;     PrimitiveDataType PrimitiveDataType;     UINT16 Size;     UINT32 Offset;     UINT8 Bit;     UINT8 DimensionCount;     UINT32 Index;     UINT32 ParentIndex;     SDimension Dimension[DTAG_ARRAY_DIMENSION]; };         </pre>
Member	<ul style="list-style-type: none"> <li>• <code>WCHAR Name[DTAG_NAME_MAX_LENGTH]</code>:  <b>The name of the tag</b></li> <li>• <code>EArea area</code>:  <b>The CPU area where the tag is located.</b></li> <li>• <code>EDataType DataType</code>:  <b>The CPU data type of the tag</b></li> <li>• <code>EPrimitiveDataType PrimitiveDataType</code>:  <b>The primitive data type of the tag</b></li> <li>• <code>UINT16 size</code>:  <b>The size of the tag in bytes</b></li> <li>• <code>UINT32 offset</code>:  <b>The byte offset of the tag if it is not located in a data block.</b></li> <li>• <code>UINT8 bit</code>:  <b>The bit offset of the tag if it is not located in a data block.</b></li> <li>• <code>UINT8 DimensionCount</code>:  <b>The number of dimensions of the array. 0 if it is not a field for the tag.</b></li> <li>• <code>UINT32 index</code>:  <b>The index of the tag</b></li> <li>• <code>UINT32 ParentIndex</code>:  <b>If this tag is embedded in another tag (for example, an element of a structure), this value then displays the index of the parent tag. The value is 0 if the tag has no parent tag.</b></li> <li>• <code>SDimension Dimension[DTAG_ARRAY_DIMENSION]</code>:  <b>Information about each dimension of the field</b></li> </ul>



Table 7- 337 STagInfo - .NET (C#)

Syntax	<pre>public struct STagInfo {     String Name;     EArea Area;     EDataType DataType;     EPrimitiveDataType PrimitiveDataType;     UInt16 Size;     UInt32 Offset;     UInt8 Bit;     UInt32 Index;     UInt32 ParentIndex;     SDimension[] Dimension; }</pre>
Member	<ul style="list-style-type: none"> <li>• String name: The name of the tag</li> <li>• EArea area: The CPU area where the tag is located.</li> <li>• EDataType DataType: The CPU data type of the tag</li> <li>• EPrimitiveDataType PrimitiveDataType: The primitive data type of the tag</li> <li>• UInt16 size: The size of the tag in bytes.</li> <li>• UInt32 offset: The byte offset of the tag if it is not located in a data block.</li> <li>• UInt8 bit: The bit offset of the tag if it is not located in a data block.</li> <li>• UInt32 index: The index of the tag</li> <li>• UInt32 ParentIndex: If this tag is embedded in another tag (for example, an element of a structure), this value then displays the index of the parent tag. The value is 0 if the tag has no parent tag.</li> <li>• SDimension[] Dimension: Information about each dimension of the field. Empty, if the tag is not an array.</li> </ul>

### 7.8.6.8 SIP

#### Description

This structure contains an IPv4 address.

Table 7- 338 SIP - .NET (C#)

Syntax	<pre>struct SIP {   byte[] IPArray { get; set; }   UInt32 IPDWord { get; set; }   string IPString { get; set; } }</pre>
Member	<ul style="list-style-type: none"><li>• UInt32 IPDWord: The IP address in a single DWORD</li><li>• byte[] IPArray: The four elements of IP in descending order</li><li>• string IPString: The IPv4 address as a string</li></ul>
Example	<p>Example for an IP address: 192.168.0.1</p> <p>SIP.IPDWord = 0xC0A80001</p> <p>SIP.IPArray[3] = 192, SIP.IPArray[2] = 168, SIP.IPArray[1] = 0, SIP.IPArray[0] = 1</p> <p>SIP.IPString = "192.168.0.1"</p>

### 7.8.6.9 SIPSuite4

#### Description

This structure contains an IPv4 suite.

Table 7- 339 SIPSuite4 - Native C++

Syntax	<pre>struct SIPSuite4 {   UIP IPAddress;   UIP SubnetMask;   UIP DefaultGateway; };</pre>
Member	<ul style="list-style-type: none"> <li>• UIP IPAddress: The IP address</li> <li>• UIP SubnetMask: The subnet mask</li> <li>• UIP DefaultGateway: The standard gateway</li> </ul>

Table 7- 340 SIPSuite4 - .NET (C#)

Syntax	<pre>struct SIPSuite4 {   SIP IPAddress;   SIP SubnetMask;   SIP DefaultGateway; }</pre>
Member	<ul style="list-style-type: none"> <li>• SIP IPAddress: The IP address</li> <li>• SIP SubnetMask: The subnet mask</li> <li>• SIP DefaultGateway: The standard gateway</li> </ul>

## 7.8.7 Enumerations

### 7.8.7.1 ERuntimeErrorCode

#### Description

This enumeration contains all error codes that are used by the Simulation Runtime API. Most API functions return one of these error codes. If the function is successful, the return value is always `SREC_OK`. Errors are returned with negative values, and alarms with positive values.

Table 7- 341 ERuntimeErrorCode - Native C++

Syntax	<pre> enum ERuntimeErrorCode {     SREC_OK = 0,     SREC_INVALID_ERROR_CODE = -1,     SREC_NOT_IMPLEMENTED = -2,     SREC_INDEX_OUT_OF_RANGE = -3,     SREC_DOES_NOT_EXIST = -4,     SREC_ALREADY_EXISTS = -5,     SREC_UNKNOWN_MESSAGE_TYPE = -6,     SREC_INVALID_MESSAGE_ID = -7,     SREC_WRONG_ARGUMENT = -8,     SREC_WRONG_PIPE = -9,     SREC_CONNECTION_ERROR = -10,     SREC_TIMEOUT = -11,     SREC_MESSAGE_CORRUPT = -12,     SREC_WRONG_VERSION = -13,     SREC_INSTANCE_NOT_RUNNING = -14,     SREC_INTERFACE_REMOVED = -15,     SREC_SHARED_MEMORY_NOT_INITIALIZED = -16,     SREC_API_NOT_INITIALIZED = -17,     SREC_WARNING_ALREADY_EXISTS = 18,     SREC_NOT_SUPPORTED = -19,     SREC_WARNING_INVALID_CALL = 20,     SREC_ERROR_LOADING_DLL = -21,     SREC_SIGNAL_NAME_DOES_NOT_EXIST = -22,     SREC_SIGNAL_TYPE_MISMATCH = -23,     SREC_SIGNAL_CONFIGURATION_ERROR = -24,     SREC_NO_SIGNAL_CONFIGURATION_LOADED = -25,     SREC_CONFIGURED_CONNECTION_NOT_FOUND = -26,     SREC_CONFIGURED_DEVICE_NOT_FOUND = -27,     SREC_INVALID_CONFIGURATION = -28,     SREC_TYPE_MISMATCH = -29,     SREC_LICENSE_NOT_FOUND = -30,     SREC_NO_LICENSE_AVAILABLE = -31,     SREC_WRONG_COMMUNICATION_INTERFACE = -32,     SREC_LIMIT_REACHED = -33,     SREC_NO_STORAGE_PATH_SET = -34,     SREC_STORAGE_PATH_ALREADY_IN_USE = -35,     SREC_MESSAGE_INCOMPLETE = -36,     SREC_ARCHIVE_STORAGE_NOT_CREATED = -37,     SREC_RETRIEVE_STORAGE_FAILURE = -38,     SREC_INVALID_OPERATING_STATE = -39,     SREC_INVALID_ARCHIVE_PATH = -40,     SREC_DELETE_EXISTING_STORAGE_FAILED = -41,     SREC_CREATE_DIRECTORIES_FAILED = -42,     SREC_NOT_ENOUGH_MEMORY = -43,     SREC_WARNING_TRIAL_MODE_ACTIVE = 44,     SREC_NOT_RUNNING = -45,     SREC_NOT_EMPTY = -46,     SREC_NOT_UP_TO_DATE = -47,     SREC_COMMUNICATION_INTERFACE_NOT_AVAILABLE = -48,     SREC_WARNING_NOT_COMPLETE = 49,     SREC_VIRTUAL_SWITCH_MISCONFIGURED = -50 };         </pre>
--------	---

Table 7- 342 ERuntimeErrorCode - .NET (C#)

Syntax	<pre> enum ERuntimeErrorCode {     OK = 0,     InvalidErrorCode = -1,     NotImplemented = -2,     IndexOutOfRange = -3,     DoesNotExist = -4,     AlreadyExists = -5,     UnknownMessageType = -6,     InvalidMessageId = -7,     WrongArgument = -8,     WrongPipe = -9,     ConnectionError = -10,     Timeout = -11,     MessageCorrupt = -12,     WrongVersion = -13,     InstanceNotRunning = -14,     InterfaceRemoved = -15,     SharedMemoryNotInitialized = -16,     ApiNotInitialized = -17,     WarningAlreadyExists = 18,     NotSupported = -19,     WarningInvalidCall = 20,     ErrorLoadingDll = -21,     SignalNameDoesNotExist = -22,     SignalTypeMismatch = -23,     SignalConfigurationError = -24,     NoSignalConfigurationLoaded = -25,     ConfiguredConnectionNotFound = -26,     ConfiguredDeviceNotFound = -27,     InvalidConfiguration = -28,     TypeMismatch = -29,     LicenseNotFound = -30,     NoLicenseAvailable = -31,     WrongCommunicationInterface = -32,     LimitReached = -33,     NoStartupPathSet = -34, <b>start</b> StartupPathAlreadyInUse = -35,     MessageIncomplete = -36, <b>archive</b> ArchiveStorageNotCreated = -37,     RetrieveStorageFailure = -38,     InvalidOperatingState = -39,     InvalidArchivePath = -40,     DeleteExistingStorageFailed = -41,     CreateDirectoriesFailed = -42,     NotEnoughMemory = -43,     WarningTrialModeActive = 44,     NotRunning = -45,     NotEmpty = -46,     NotUpToDate = -47,     CommunicationInterfaceNotAvailable = -48,     WarningNotComplete = 49,     VirtualSwitchMisconfigured = -50 } </pre>
--------	---

### 7.8.7.2 EArea

#### Description

This list contains all PLC areas that contain the available PLC tags.

Table 7- 343 EArea - Native C++

Syntax	<pre>enum EArea {     SRA_INVALID_AREA = 0,     SRA_INPUT = 1,     SRA_MARKER = 2,     SRA_OUTPUT = 3,     SRA_COUNTER = 4,     SRA_TIMER = 5,     SRA_DATABLOCK = 6,      SRA_ENUMERATION_SIZE = 7 };</pre>
--------	--

Table 7- 344 EArea - .NET (C#)

Syntax	<pre>public enum EArea {     InvalidArea = 0,     Input = 1,     Marker = 2,     Output = 3,     Counter = 4,     Timer = 5,     DataBlock = 6, }</pre>
--------	---

### 7.8.7.3 EOperatingState

#### Description

This list contains all the operating states of a virtual controller.

Table 7- 345 EOperatingState - Native C++

Syntax	<pre>enum EOperatingState {     SROS_INVALID_OPERATING_STATE = 0,     SROS_OFF = 1,     SROS_BOOTING = 2,     SROS_STOP = 3,     SROS_STARTUP = 4,     SROS_RUN = 5,     SROS_FREEZE = 6,     SROS_SHUTTING_DOWN = 7,      SROS_ENUMERATION_SIZE = 8 };</pre>
--------	---

Table 7- 346 EOperatingState - .NET (C#)

Syntax	<pre>enum EOperatingState {     InvalidOperatingState = 0,     Off = 1,     Booting = 2,     Stop = 3,     Startup = 4,     Run = 5,     Freeze = 6,     ShuttingDown = 7 }</pre>
--------	---

### 7.8.7.4 EOperatingMode

#### Description

This list contains all the operating modes of a virtual controller.

Table 7- 347 EOperatingMode - Native C++

Syntax	<pre>enum EOperatingMode {     SROM_DEFAULT = 0,     SROM_SINGLE_STEP = 1,     SROM_EXTENDED_SINGLE_STEP = 2,     SROM_TIMESPAN_SYNCHRONIZED = 3 };</pre>
--------	---

Table 7- 348 EOperatingMode - .NET (C#)

Syntax	<pre>enum EOperatingMode {     Default = 0,     SingleStep = 1,     ExtendedSingleStep = 2,     TimespanSynchronized = 3 };</pre>
--------	---



### 7.8.7.5 ECPUType

#### Description

This list contains all CPU types that can be loaded in a virtual controller.

Table 7- 349 ECPUType - Native C++

Syntax	<pre>enum ECPUType {   SRCT_1500_Unspecified = 0x000005DC,   SRCT_1511 = 0x000005E7,   SRCT_1511v2 = 0x010005E7,   SRCT_1513 = 0x000005E9,   SRCT_1513v2 = 0x010005E9,   SRCT_1515 = 0x000005EB,   SRCT_1515v2 = 0x010005EB,   SRCT_1516 = 0x000005EC,   SRCT_1516v2 = 0x010005EC,   SRCT_1517 = 0x000005ED,   SRCT_1518 = 0x000005EE,   SRCT_1511C = 0x000405E7,   SRCT_1512C = 0x000405E8,   SRCT_1511F = 0x000105E7,   SRCT_1511Fv2 = 0x010105E7,   SRCT_1513F = 0x000105E9,   SRCT_1513Fv2 = 0x010105E9,   SRCT_1515F = 0x000105EB,   SRCT_1515Fv2 = 0x010105EB,   SRCT_1516F = 0x000105EC,   SRCT_1516Fv2 = 0x010105EC,   SRCT_1517F = 0x000105ED,   SRCT_1518F = 0x000105EE,   SRCT_1511T = 0x000805E7,   SRCT_1515T = 0x000805EB,   SRCT_1517T = 0x000805ED,   SRCT_1517TF = 0x000905ED,   SRCT_1518ODK = 0x001005EE,   SRCT_1518FODK = 0x001105EE,   SRCT_ET200SP_Unspecified = 0x000205DC,   SRCT_1510SP = 0x000205E6,   SRCT_1510SPv2 = 0x010205E6,   SRCT_1512SP = 0x000205E8,   SRCT_1512SPv2 = 0x010205E8,   SRCT_1510SPF = 0x000305E6,   SRCT_1510SPFv2 = 0x010305E6,   SRCT_1512SPF = 0x000305E8,   SRCT_1512SPFv2 = 0x010305E8 };</pre>
--------	---

Table 7- 350 ECPUType - .NET (C#)

Syntax	<pre> enum ECPUType {     CPU1500_Unspecified = 0x000005DC,     CPU1511 = 0x000005E7,     CPU1511v2 = 0x010005E7,     CPU1513 = 0x000005E9,     CPU1513v2 = 0x010005E9,     CPU1515 = 0x000005EB,     CPU1515v2 = 0x010005EB,     CPU1516 = 0x000005EC,     CPU1516v2 = 0x010005EC,     CPU1517 = 0x000005ED,     CPU1518 = 0x000005EE,     CPU1511C = 0x000405E7,     CPU1512C = 0x000405E8,     CPU1511F = 0x000105E7,     CPU1511Fv2 = 0x010105E7,     CPU1513F = 0x000105E9,     CPU1513Fv2 = 0x010105E9,     CPU1515F = 0x000105EB,     CPU1515Fv2 = 0x010105EB,     CPU1516F = 0x000105EC,     CPU1516Fv2 = 0x010105EC,     CPU1517F = 0x000105ED,     CPU1518F = 0x000105EE,     CPU1511T = 0x000805E7,     CPU1515T = 0x000805EB,     CPU1517T = 0x000805ED,     CPU1517TF = 0x000905ED,     CPU1518ODK = 0x001005EE,     CPU1518FODK = 0x001105EE,     CPUET200SP_Unspecified = 0x000205DC,     CPU1510SP = 0x000205E6,     CPU1510SPv2 = 0x010205E6,     CPU1512SP = 0x000205E8,     CPU1512SPv2 = 0x010205E8,     CPU1510SPF = 0x000305E6,     CPU1510SPFv2 = 0x010305E6,     CPU1512SPF = 0x000305E8,     CPU1512SPFv2 = 0x010305E8 }                 </pre>
--------	--

### 7.8.7.6 ECommunicationInterface

#### Description

This list shows the available communication interfaces of a virtual controller.

Table 7- 351 ECommunicationInterface - Native C++

Syntax	<pre>enum ECommunicationInterface {     SRCI_NONE = 0,     SRCI_SOFTBUS = 1,     SRCI_TCPIP = 2,     SRCI_ENUMERATION_SIZE = 3 };</pre>
--------	---

Table 7- 352 ECommunicationInterface - .NET (C#)

Syntax	<pre>enum ECommunicationInterface {     None = 0,     Softbus = 1,     TCPIP = 2, }</pre>
--------	---

### 7.8.7.7 ELEDType

#### Description

This list includes all types of LEDs of a virtual controller.

Table 7- 353 ELEDType - Native C++

Syntax	<pre>enum ELEDType {     SRLT_STOP = 0,     SRLT_RUN = 1,     SRLT_ERROR = 2,     SRLT_MAINT = 3,     SRLT_REDUND = 4,     SRLT_FORCE = 5,     SRLT_BUSF1 = 6,     SRLT_BUSF2 = 7,     SRLT_BUSF3 = 8,     SRLT_BUSF4 = 9,      SRLT_ENUMERATION_SIZE = 10 };</pre>
--------	---

Table 7- 354 ELEDType - .NET (C#)

Syntax	<pre>enum ELEDType {     Stop = 0,     Run = 1,     Error = 2,     Maint = 3,     Redund = 4,     Force = 5,     Busf1 = 6,     Busf2 = 7,     Busf3 = 8,     Busf4 = 9, }</pre>
--------	--

### 7.8.7.8 ELEDMode

#### Description

This list contains all the LED states of a virtual controller.

Table 7- 355 ELEDMode - Native C++

Syntax	<pre>enum ELEDMode {     SRLM_OFF = 0,     SRLM_ON = 1,     SRLM_FLASH_FAST = 2,     SRLM_FLASH_SLOW = 3,     SRLM_INVALID = 4 };</pre>
--------	---

Table 7- 356 ELEDMode - .NET (C#)

Syntax	<pre>enum ELEDMode {     Off = 0,     On = 1,     FlashFast = 2,     FlashSlow = 3,     Invalid = 4 };</pre>
--------	--

### 7.8.7.9 EPrimitiveDataType

#### Description

This list contains all the primitive data types that are used by the I/O access functions.

Table 7- 357 EPrimitiveDataType - Native C++

Syntax	<pre>enum EPrimitiveDataType {     SRPDT_UNSPECIFIC = 0,     SRPDT_STRUCT = 1,     SRPDT_BOOL = 2,     SRPDT_INT8 = 3,     SRPDT_INT16 = 4,     SRPDT_INT32 = 5,     SRPDT_INT64 = 6,     SRPDT_UINT8 = 7,     SRPDT_UINT16 = 8,     SRPDT_UINT32 = 9,     SRPDT_UINT64 = 10,     SRPDT_FLOAT = 11,     SRPDT_DOUBLE = 12,     SRPDT_CHAR = 13,     SRPDT_WCHAR = 14 };</pre>
--------	---

Table 7- 358 EPrimitiveDataType - .NET (C#)

Syntax	<pre>enum EPrimitiveDataType {     Unspecific = 0,     Struct = 1,     Bool = 2,     Int8 = 3,     Int16 = 4,     Int32 = 5,     Int64 = 6,     UInt8 = 7,     UInt16 = 8,     UInt32 = 9,     UInt64 = 10,     Float = 11,     Double = 12,     Char = 13,     WChar = 14 };</pre>
--------	---

## Compatible primitive data types

The following tables shows the primitive data types of the user interface (API) and the data types of the PLCSIM Advanced instance that are configured in the stored tag list. The data types that can be used as compatible are marked with "X".

Table 7- 359 Compatible primitive data types - Reading

API	PLCSIM Advanced instance												
	Bool	INT8	INT16	INT32	INT64	UINT8	UINT16	UINT32	UINT64	Float	Double	Char	WChar
Bool	X												
INT8		X											
INT16		X	X			X							
INT32		X	X	X		X	X						
INT64		X	X	X	X	X	X	X					
UINT8						X							
UINT16						X	X						
UINT32						X	X	X					
UINT64						X	X	X	X				
Float										X			
Double											X		
Char												X	
WChar													X

Table 7- 360 Compatible primitive data types - Write

API	PLCSIM Advanced instance												
	Bool	INT8	INT16	INT32	INT64	UINT8	UINT16	UINT32	UINT64	Float	Double	Char	WChar
Bool	X												
INT8		X	X	X	X								
INT16			X	X	X								
INT32				X	X								
INT64					X								
UINT8			X	X	X	X	X	X	X				
UINT16							X	X	X				
UINT32								X	X				
UINT64									X				
Float										X			
Double											X		
Char												X	
WChar													X

### **7.8.7.10 EDataType**

#### **Description**

This list contains all the CPU data types (STEP 7).



Table 7- 361 EDataType - Native C++

Syntax	<pre> enum EDataType {   SRDT_UNKNOWN = 0,   SRDT_BOOL = 1,   SRDT_BYTE = 2,   SRDT_CHAR = 3,   SRDT_WORD = 4,   SRDT_INT = 5,   SRDT_DWORD = 6,   SRDT_DINT = 7,   SRDT_REAL = 8,   SRDT_DATE = 9,   SRDT_TIME_OF_DAY = 10,   SRDT_TIME = 11,   SRDT_S5TIME = 12,   SRDT_DATE_AND_TIME = 14,   SRDT_STRUCT = 17,   SRDT_STRING = 19,   SRDT_COUNTER = 28,   SRDT_TIMER = 29,   SRDT_IEC_Counter = 30,   SRDT_IEC_Timer = 31,   SRDT_LREAL = 48,   SRDT_ULINT = 49,   SRDT_LINT = 50,   SRDT_LWORD = 51,   SRDT_USINT = 52,   SRDT_UINT = 53,   SRDT_UDINT = 54,   SRDT_SINT = 55,   SRDT_WCHAR = 61,   SRDT_WSTRING = 62,   SRDT_LTIME = 64,   SRDT_LTIME_OF_DAY = 65,   SRDT_LDT = 66,   SRDT_DTL = 67,   SRDT_IEC_LTimer = 68,   SRDT_IEC_SCounter = 69,   SRDT_IEC_DCounter = 70,   SRDT_IEC_LCounter = 71,   SRDT_IEC_UCounter = 72,   SRDT_IEC_USCounter = 73,   SRDT_IEC_UDCounter = 74,   SRDT_IEC_ULCounter = 75,   SRDT_ERROR_STRUCT = 97,   SRDT_NREF = 98,   SRDT_CREF = 101,   SRDT_AOM_IDENT = 128,   SRDT_EVENT_ANY = 129,   SRDT_EVENT_ATT = 130,   SRDT_EVENT_HWINT = 131,   SRDT_HW_ANY = 144,   SRDT_HW_IOSYSTEM = 145,   SRDT_HW_DPMASTER = 146,   SRDT_HW_DEVICE = 147,   SRDT_HW_DP_SLAVE = 148,   SRDT_HW_IO = 149,   SRDT_HW_MODULE = 150,   SRDT_HW_SUBMODULE = 151,   SRDT_HW_HSC = 152,   SRDT_HW_PWM = 153,   SRDT_HW_PTO = 154,   SRDT_HW_INTERFACE = 155,   SRDT_HW_IEPORT = 156,   SRDT_OB_ANY = 160,   SRDT_OB_DELAY = 161,   SRDT_OB_TOD = 162,   SRDT_OB_CYCLIC = 163,   SRDT_OB_ATT = 164,   SRDT_CONN_ANY = 168,   SRDT_CONN_PRG = 169,   SRDT_CONN_OUC = 170,   SRDT_CONN_R_ID = 171,   SRDT_PORT = 173,   SRDT_RTM = 174,   SRDT_PIP = 175,   SRDT_OB_PCYCLE = 192,   SRDT_OB_HWINT = 193,   SRDT_OB_DIAG = 195,   SRDT_OB_TIMEERROR = 196,   SRDT_OB_STARTUP = 197,   SRDT_DB_ANY = 208,   SRDT_DB_WWW = 209,   SRDT_DB_DYN = 210,   SRDT_DB_ = 257 }; </pre>
--------	--

Table 7- 362 EDataType - .NET (C#)

Syntax	<pre> public enum EDataType {     Unknown = 0,     Bool = 1,     Byte = 2,     Char = 3,     Word = 4,     Int = 5,     DWord = 6,     DInt = 7,     Real = 8,     Date = 9,     TimeOfDay = 10,     Time = 11,     S5Time = 12,     DateAndTime = 14,     Struct = 17,     String = 19,     Counter = 28,     Timer = 29,     IEC_Counter = 30,     IEC_Timer = 31,     LReal = 48,     ULInt = 49,     LInt = 50,     LWord = 51,     USInt = 52,     UInt = 53,     UDInt = 54,     Sint = 55,     WChar = 61,     WString = 62,     LTime = 64,     LTimeOfDay = 65,     LDT = 66,     DTL = 67,     IEC_LTimer = 68,     IEC_SCounter = 69,     IEC_DCounter = 70,     IEC_LCounter = 71,     IEC_UCounter = 72,     IEC_USCounter = 73,     IEC_UDCounte = 74,     IEC_ULCounter = 75,     ErrorStruct = 97,     NREF = 98,     CREF = 101,     Aom Ident = 128,     Event Any = 129,     Event_Att = 130,     Event_HwInt = 131,     Hw Any = 144,     Hw_IoSystem = 145,     Hw_DpMaster = 146,     Hw_Device = 147,     Hw_DpSlave = 148,     Hw_Io = 149,     Hw_Module = 150,     Hw_SubModule = 151,     Hw_Hsc = 152,     Hw_Pwm = 153,     Hw_Pto = 154,     Hw_Interface = 155,     Hw_IEPort = 156,     OB_Any = 160,     OB_Delay = 161,     OB_Tod = 162,     OB_Cyclic = 163,     OB_Att = 164,     Conn Any = 168,     Conn_Prg = 169,     Conn_Ouc = 170,     Conn_R_ID = 171,     Port = 173,     Rtm = 174,     Pip = 175,     OB_PCycle = 192,     OB_HwInt = 193,     OB_Diag = 195,     OB_TimeError = 196,     OB_Startup = 197,     DB_Any = 208,     DB_WWW = 209,     DB_Dyn = 210,     DB = 257 }                 </pre>
--------	---

### 7.8.7.11 ETagListDetails

#### Description

This list contains all PLC areas that can be used as a filter to update the tag table.

Table 7- 363 ETagListDetails - Native C++

Syntax	<pre>enum ETagListDetails {     SRTLD_NONE = 0,     SRTLD_IO = 1,     SRTLD_M = 2,     SRTLD_IOM = 3,     SRTLD_CT = 4,     SRTLD_IOCT = 5,     SRTLD_MCT = 6,     SRTLD_IOMCT = 7,     SRTLD_DB = 8,     SRTLD_IODB = 9,     SRTLD_MDB = 10,     SRTLD_IOMDB = 11,     SRTLD_CTDB = 12,     SRTLD_IOCTDB = 13,     SRTLD_MCTDB = 14,     SRTLD_IOMCTDB = 15 };</pre>
--------	---

Table 7- 364 ETagListDetails - .NET (C#)

Syntax	<pre>enum ETagListDetails {     None = 0,     IO = 1,     M = 2,     IOM = 3,     CT = 4,     IOCT = 5,     MCT = 6,     IOMCT = 7,     DB = 8,     IODB = 9,     MDB = 10,     IOMDB = 11,     CTDB = 12,     IOCTDB = 13,     MCTDB = 14,     IOMCTDB = 15 };</pre>
--------	---

### 7.8.7.12 ERuntimeConfigChanged

#### Description

This list contains all possible causes of a OnConfigurationChanged event that the Runtime Manager sends.

Table 7- 365 ERuntimeConfigChanged - Native C++

Syntax	<pre>enum ERuntimeConfigChanged {     SRCC_INSTANCE_REGISTERED = 0,     SRCC_INSTANCE_UNREGISTERED = 1,     SRCC_CONNECTION_OPENED = 2,     SRCC_CONNECTION_CLOSED = 3,     SRCC_PORT_OPENED = 4,     SRCC_PORT_CLOSED = 5 };</pre>
--------	---

Table 7- 366 ERuntimeConfigChanged - .NET (C#)

Syntax	<pre>enum ERuntimeConfigChanged {     InstanceRegistered = 0,     InstanceUnregistered = 1,     ConnectionOpened = 2,     ConnectionClosed = 3,     PortOpened = 4,     PortClosed = 5 };</pre>
--------	---

### 7.8.7.13 EInstanceConfigChanged

#### Description

This list contains all possible causes for a OnConfigurationChanged event that the virtual controller sends.

Table 7- 367 EInstanceConfigChanged - Native C++

Syntax	<pre>enum EInstanceConfigChanged {     SRICC_HARDWARE_SOFTWARE_CHANGED = 0,     SRICC_IP_CHANGED = 1 };</pre>
--------	---

Table 7- 368 EInstanceConfigChanged - .NET (C#)

Syntax	<pre>enum EInstanceConfigChanged {     HardwareSoftwareChanged = 0,     IPChanged = 1 };</pre>
--------	--



# Restrictions

## 8.1 Overview

Certain actions or events may lead to behavior in S7-PLCSIM Advanced or in STEP 7 which deviates from that of a hardware CPU. Restrictions and possible remedies can be found in the following sections:

- OPC UA server (Page 302)
- Web server (Page 304)
- Restrictions for communications services (Page 305)
- Restrictions for instructions (Page 306)
- Restrictions with Motion Control (Page 307)
- Restrictions to local communication via Softbus (Page 308)
- Error with overflow cyclical events (Page 309)
- Deviating I/O values in the STEP 7 user program (Page 310)
- Multiple simulations and possible collision of IP addresses (Page 310)
- Simulation in standby mode (Page 310)
- Error installing the antivirus software from Kaspersky (Page 311)

## 8.2 OPC UA server

With OPC UA, data exchange is performed through an open, standardized and manufacturer-independent communication protocol. The CPU acting as the OPC UA server can communicate with OPC UA clients, for example, with HMI panels V14 and SCADA systems.

For technical reasons, the security settings in PLCSIM Advanced differ from a hardware CPU. Some features are disabled for simulations or are available to a limited extent.

### Configuring OPC UA server

Start the instances via the communication interface "PLCSIM Virtual Ethernet Adapter" (TCP/IP) to use the OPC UA server.

The OPC UA server functionality is not available if communication takes place via the Softbus.

## OPC UA security settings

Based on the OPC UA security settings, the same settings can be made in STEP 7 for the hardware CPU. The data is not further processed by PLCSIM Advanced. This ensures that the user does not have his project change to perform a simulation.

## Certificate

- **Server certificate**

PLCSIM Advanced uses its own certificate in the firmware and not that of STEP 7. The certificate need not be changed for simulations. However, it does not have the same security level as a downloaded server certificate and cannot be used for secure connections!

- **Server Security Endpoints**

PLCSIM Advanced only supports Security Endpoint "none".

- **Client certificate**

PLCSIM Advanced does not evaluate certificates the imported and configured in STEP 7 . PLCSIM Advanced accepts all client certificates automatically. This setting cannot be changed.

- **User authentication**

PLCSIM Advanced does not use the user name configured in STEP 7.

Only one logon is possible as "guest" or "anonymous".

## 8.3 Web server

The Web server integrated in a CPU enables monitoring and administering of the CPU by authorized users over a network. This permits evaluation and diagnostics over long distances.

The simulation of the Web server is restricted under S7-PLCSIM V1.0 Advanced.

Each PLCSIM Advanced instance can simulate its own Web server.

The freeze state of a virtual controller is not shown as an internal operating state.

### Configuring the Web server

#### S7 PLCSIM Advanced

Start the instances via the communication interface "PLCSIM Virtual Ethernet Adapter" (TCP/IP) to use the Web server.

The Web server functionality is not available if the communication is performed via the Softbus.

#### STEP 7

Configure the Web server in STEP 7 in the CPU properties.

### Restricted Web server functionality

- **Logon**  
"PLCSIM" is preset as the user. There is no logon for users.  
A user configured in STEP 7 and his rights have no effect on the "PLCSIM" user.
- There is no access via the secure transmission protocol "HTTPS".
- The information may not be fully displayed on some websites due to different data handling.
- There is no topology information.
- "Online Backup&Restore" is not available.
- FW updates are not supported.



## 8.4 Restrictions for communications services

### TUSEND / TURCV

When you run the UDP blocks TUSEND and TURCV via the "PLCSIM" communication interface (Softbus), you get error code 0x80C4 at the transmission end and receiving end:

Temporary communications error. The specified connection is temporarily down.

#### **Remedy**

Set "PLCSIM Virtual Ethernet Adapter" (TCP/IP) as the communication interface in PLCSIM Advanced.

## 8.5 Restrictions for instructions

S7-PLCSIM Advanced simulates instructions for CPUs S7-1500 and ET 200SP with a few exceptions, such as interrupts.

Some instructions are partially supported. For these, S7-PLCSIM Advanced checks the input parameters for validity and returns outputs that are valid but do not necessarily correspond to those that a real CPU with physical inputs/outputs would return.

### Instructions not supported

Unsupported instructions are handled as not ready by S7-PLCSIM Advanced, their value is always "OK".

S7PLCSIM Advanced does not support the following instructions:

Table 8- 1 Instructions not supported

Instruction	Description
DIS_AIRT	Delay interrupts with higher priority and asynchronous errors
EN_AIRT	Activate interrupts with higher priority and asynchronous errors
DIS_IRT	Call new interrupts and disable asynchronous error
RE_TRIGR	Start time monitoring
WR_DPARM	Transfer data record
GETIO	Read process image
SETIO	Transfer process image
GETIO_PART	Read a part of the outputs of a DP standard slave / PROFINET IO device
SETIO_PART	Write a part of the outputs of a DP standard slave / PROFINET IO device
DPRD_DAT	Read consistent data of a standard DP slave
DPWR_DAT	Write consistent data to a standard DP slave
RD_OBINF	Read OB start information
DP_TOPOLOG	DP bus topology
PORT_CFG	Safety function
ATTACH	Assigning a hardware interrupt to an OB
DETACH	Removing the mapping between hardware interrupt and OB

## 8.6 Restrictions with Motion Control

### 8.6.1 Motion Control resources

There are motion control resources on each CPU that can be distributed to the technology objects.

---

#### Note

##### Maximum 5120 Motion Control resources

PLCSIM Advanced supports a maximum of 5120 motion control resources. The simulation of motion control is therefore restricted for the following CPUs:

---

Table 8- 2 CPUs with limited Motion Control resources

Type		Article number
Standard CPUs	CPU 1517-3 PN/DP	6ES7517-3AP00-0AB0
	CPU 1518-4 PN/DP	6ES7518-4AP00-0AB0
	CPU 1518-4 PN/DP ODK <sup>1</sup>	6ES7518-4AP00-3AB0
Fail-safe CPUs	CPU 1517F-3 PN/DP	6ES7517-3FP00-0AB0
	CPU 1518F-4 PN/DP	6ES7518-4FP00-0AB0
	CPU 1518F-4 PN/DP ODK <sup>1</sup>	6ES7518-4FP00-3AB0
Technology CPUs	CPU 1517T-3 PN/DP <sup>2</sup>	6ES7517-3TP00-0AB0
	CPU 1517TF-3 PN/DP <sup>2</sup>	6ES7517-3UP00-0AB0

<sup>1</sup> The ODK functionality of this CPU is not simulated.

<sup>2</sup> The simulation of this CPU supports only 64 cams.

### Exceeding the quantity structure

STEP 7 checks whether the quantity structure for a configured CPU is complied with, and alerts you when it is exceeded. When you load a project into a CPU, another message appears.

---

#### Note

A project that uses more than 5120 motion control resources can be downloaded to a virtual controller. However, there is no message indicating that the quantity structure has been exceeded.

Only when the user program accesses the technology objects do you recognize by the error message at the block and the value 0 of the object that motion control objects cannot be used due to an overrun of the quantity structure.

---

## 8.6.2 Technology modules

PLCSIM Advanced simulates the real CPU, but not configured, connected technology modules or other I/O devices.

It is possible to download a project with technology modules for operation of motion control. However, the built-in logic of the technology modules is not part of the simulation, which is why the associated Motion Control instructions are not supported.

### Additional information

For more information on Motion Control, refer to the following manuals:

- Device manuals to support SIMATIC controllers (<http://w3.siemens.com/mcims/industrial-automation-systems-simatic/en/manual-overview/Pages/Default.aspx>)
- Function manual S7-1500 Motion Control (<https://support.industry.siemens.com/cs/ww/en/view/109739589>)
- Function manual S7-1500T Motion Control (<https://support.industry.siemens.com/cs/ww/en/view/109481326>)

## 8.7 Restrictions to local communication via Softbus

### Identical IP addresses for instances

If the "PLCSIM" communication interface (Softbus) is set, then identical IP addresses are created automatically for all instances when creating the instances through the Control Panel.

In STEP 7, only one instance is therefore displayed in the lifelist.

#### Remedy

Use the API function `SetIPSuite()` to assign a unique address for each instance, then all instances are displayed in STEP 7 with their IP addresses.

#### API function

- `SetIPSuite()` (Page 122)

### Online and diagnostics

If the "PLCSIM" (Softbus) communication interface is set, no details are displayed for the "Online and Diagnostics" function under the PROFINET interface (IP address, MAC address, etc.).

### See also

Controller - Information and settings (Page 119)

## 8.8 Restrictions of security with VMware vSphere Hypervisor (ESXi)

When you use the virtualization platform VMware vSphere Hypervisor (ESXi), you must change the policy exception to communicate over TCP/IP.

### Remedy

Accept the "Promiscuous mode and "Forged transmit" options for the Virtual Switch of the ESXi.

#### NOTICE

##### Restrictions of security

For security reasons, Promiscuous mode is disabled by default.

If you accept the Promiscuous mode, the real Ethernet adapter even receives telegrams that are not addressed to it.

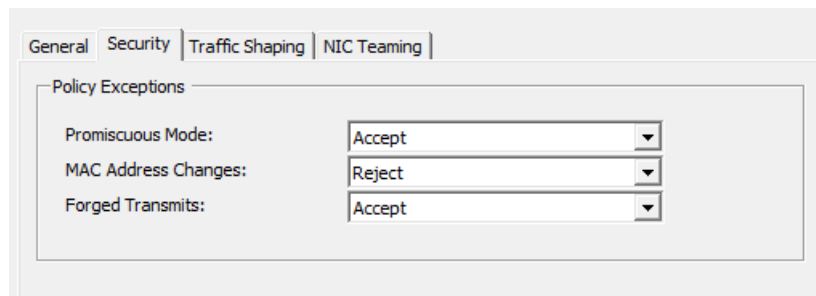


Figure 8-1 Policy exceptions for VMware vSphere Hypervisor (ESXi)

## 8.9 Error with overflow cyclical events

If your simulation contains cyclic interrupts, the queue of PLCSIM Advanced may overflow for cyclic events. Due to the execution speed of PLCSIM Advanced compared to real hardware, the time required to create the diagnostics buffer entry may be longer than the time until the next cyclic interrupt.

In this case, an additional entry is placed in the queue, causing another overflow. In the event of an overflow, PLCSIM Advanced provides visual information in the form of diagnostics buffer messages and a red error icon in the project tree.

### See also

Speed up and slow down simulation (Page 60)

## 8.10 Deviating I/O values in the STEP 7 user program

### Updated values

Each value change made by a STEP 7 user program in the I/O address areas is overwritten in the cycle control point with the updated value that was written via the API functions `Write...()`. The API functions `Read...()` only return this updated value and not the value from STEP 7 **for the input range**.

### Non-updated values

If the value was not updated via the API functions `Write...()`, the API functions `Read...()` return the value from STEP 7 **for the output range**.

### See also

Simulate peripheral I/O (Page 53)

## 8.11 Multiple simulations and possible collision of IP addresses

You can simultaneously simulate multiple CPUs, but each simulated CPU interface requires a unique IP address.

Make sure your CPUs have different IP addresses before starting the simulation.

## 8.12 Lacking access to an IP address

### Special feature of distributed communication

If you use multiple network nodes on the same subnet through different virtual or real adapters, the operating system may search for the node on the wrong adapter.

### Remedy

Repeat your requests or enter "arp -d <IP address>" in the command line editor of Windows.

## 8.13 Simulation in standby mode

If your computer or programming device goes into standby or hibernation mode, the simulation may be stopped. In this case, the communication between STEP 7 and S7-PLCSIM Advanced is stopped. When your computer or programming device starts up again, the communication may need to be reestablished. In some cases, it may also be necessary to open the simulation project again.

To prevent this situation, disable the standby mode on your computer or programming device.

## **8.14 Error installing the antivirus software from Kaspersky**

When using the Anti-Virus virus scanner from Kaspersky, network settings may not be correctly during installation of PLCSIM Advanced. The result is that communication via TCP/IP cannot be used (error code -50 in the Control Panel).

### **Remedy**

Check your network settings as described in the section [Enable distributed communication](#) (Page 43).

# List of abbreviations

# A

Abbreviation	Term
ALM	Automation License Manager Tool for managing license keys in STEP 7
API	Application Programming Interface user interface
arp	Address resolution protocol
BCD	Binary Coded Decimal
CPU	Central Processing Unit (Synonym for PLC)
ES	Engineering System
HMI	Human Machine Interface user interface
IE	Industrial Ethernet
GUI	Graphical User Interface
LAN	Local Area Network Computer network that is limited to a local area.
OB	Organization Block
ODK	Open Development Kit
OPC UA	Open Platform Communications Unified Architecture
PG	Programming device
PLC	Programmable Logic Controller
PN	PROFINET
RAM	Random Access Memory
RT	Runtime
TIA	Totally Integrated Automation
UTC	Coordinated Universal Time
VM	Virtual Machine
VPLC	Virtual Programmable Logic Controller
WinCC	Windows Control Center