

CX-Supervisor .NET Interface Reference

Software Version 3.0

© OMRON, 2009

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.

All copyright and trademarks acknowledged.

No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Warning:	Failure to read and understand the information provided in this manual may result in personal injury or death, damage to the product, or product failure. Please read each chapter in its entirety and be sure you understand the information provided in the chapter and related chapters before attempting any of the procedures or operations given.
-----------------	---

Table of Contents

Introduction.....	4
Who Should Read This Document.....	4
Interface Overview	5
Referencing the DLL	5
Examples.....	6
VB .NET	6
C# .NET	6
Class Reference	7
PointMngt.....	7
ListGroups.....	7
ListPoints.....	7
SetValue.....	7
GetValue	8
IsValidPoint	8
IsValidGroup	8
GetPointData.....	8
BrowsePoints	9
ApplicationMngt.....	10
GetErrorString	10
Restart.....	10
GetProjectName.....	10
ListDevices.....	10
GetDeviceStatus	11
GetAppld	11
IsValidAppld	11
AlarmMngt.....	12
ListAlarmGroups	12
ListAlarms	12
GetAlarmData	12
AcknowledgeAlarm.....	13
AcknowledgeAllAlarms.....	13
BrowseAlarms	13
GetAlarmLog	13
GetActiveAlarms	14
ScriptMngt.....	15
ExecuteScript.....	15
ListScripts.....	15
GetScriptParameters.....	15
ErrorMngt	16
GetErrorLog	16

Introduction

The purpose of this document is to act as an introduction and reference guide to the CX-Supervisor .NET interface introduced in V3.0.

Who Should Read This Document

The target audience is application developers knowledgeable in the .NET Framework and using external libraries. No attempt is made to teach .NET practices or any programming language.

Interface Overview

The CX-Supervisor .NET interface allow external applications to access CX-Supervisor using well defined interfaces to access specific areas of functionality. The interface can be divided in to the following areas of functionality:

- Point monitoring and data gathering
- Application/System management
- Alarm monitoring and control
- Script execution
- Error management

The reference section of this document provides full details of the classes and methods which implement this functionality.

Referencing the DLL

The interface is exposed through the file *SCSRUNLib.dll* which is located in the CX-Supervisor installation directory. In order to use the interface classes you must first add a reference to it in your project and then insert an appropriate *using* or *Imports* declaration in your class.

Examples

These code snippets are provided as an example of how to use the CX-Supervisor .NET Interface.

VB .NET

```
' declare variables
CErrorMngtClass errorMngr = new CErrorMngtClass();
object list;
' get list of errors
errorMngr.GetErrorLog(out list);
' convert list to string array
string[] errors = (string[])list;
' use error log...
```

C# .NET

```
// declare variables
CPointMngtClass pointMngr = new CPointMngtClass();
object list;
// get list of points
pointMngr.ListPoints(out list);
// convert list to string array
string[] points = (string[])list;
// use points list...
```

VB .NET and ASP.NET

The source code for the CX-Supervisor Standard Web Pages is installed along with CX-Supervisor. It can be opened and run using Visual Web Developer 2008. The Express edition of this software is available as a free download from Microsoft.

Class Reference

PointMngt

Allows the client to manage the acquisition of point information and the reading and writing of point data.

ListGroups

Retrieves a list of all the point groups contained within a CX-Supervisor application.

```
ListGroups(ByRef pGroups As Object)
```

Parameters	Description
pGroups	An array of strings representing the names of all the groups

ListPoints

Retrieves a list of all the points that are part of the given group.

```
ListPoints(ByVal szName As String, ByRef pPoints As Object)
```

Parameters	Description
szName	Name of Group
pPoints	Receives an array of strings representing the names of all the points contained in the group

SetValue

This method sets a point to a specified value. In circumstances where the point can not be set to a value (i.e. More than allowed maximum) the value returned in the retVal parameter will differ from that specified and represent the actual value the point was set to..

```
SetValue(ByVal szName As String, ByVal varValue As Object, ByRef retVal As Object)
```

Parameters	Description
szName	Name of Point
varValue	Value the point is to be set to
retVal	The actual value the point was set to

GetValue

Reads the current value of a point.

```
GetValue(ByVal szName As String, ByRef retVal As Object)
```

Parameters	Description
szName	Name of Point
retVal	Receives the current value of point

IsValidPoint

Determines whether the point is valid.

```
IsValidPoint(ByVal szName As String, ByVal szGroup As String, ByRef  
retVal As Integer)
```

Parameters	Description
szName	Name of Point
szGroup	Name of Group
retVal	Receives true or false

IsValidGroup

Determines whether the group is valid.

```
IsValidPoint(ByVal szName As String, ByRef retVal As Integer)
```

Parameters	Description
szName	Name of Group
retVal	Receives true or false

GetPointData

Retrieves the metadata of a specific point

```
GetPointData(ByVal szName As String, ByRef vartype As UShort, ByRef  
pDescription As Object, ByRef bReadOnly As Integer, ByRef iArraySize  
As Integer, ByRef Value As Object)
```

Parameters	Description
szName	Name of the Point
vartype	The type of point

pDescription	The description of the point
iArraySize	Number of array elements (1 = non array point)
value	The current value of the point

BrowsePoints

This method retrieves a filtered list of the points contained within a CX-Supervisor application.

```
BrowsePoints(ByVal szFilter As String, ByVal szGroup As String, ByVal  
vtDataTypeFilter As UShort, ByRef pPoints As Object)
```

Parameters	Description
szFilter	Free format filter eg. "P*"
szGroup	Group name filter
vtDataTypeFilter	Data type filter. One of: 0 – All types 11 - Boolean 3 - Integer 5 - Real 8 - String
pPoints	Receives an array of strings representing the names of all the points that pass the filters

ApplicationMngt

Allows the client to retrieve Application level information and manage the running application.

GetErrorString

This method returns the error string corresponding to a specific error code generated by CX-Supervisor in response to a method call on the custom interface.

```
GetErrorString(ByVal dwError As Integer, ByRef pString As Object)
```

Parameters	Description
dwError	A valid component specific error code that the client had returned from an interface function from the component. For which the client application is requesting the server's textual representation
pString	Receives a string error message

Restart

Causes the CX-Supervisor runtime to be restarted. If a path to a .SR2 file is specified then this is run upon restart.

```
Restart(ByVal szFile As String)
```

Parameters	Description
szFile	Optional path of the SCS application to run.

GetProjectName

Returns the name of the running CX-Supervisor application project..

```
GetProjectName(ByRef pName As Object)
```

Parameters	Description
pName	Receives the name of the project

ListDevices

Retrieves a list of all the devices contained within a CX-Supervisor application.

```
ListDevices(ByRef pDevices As Object)
```

Parameters	Description
pDevices	An array of strings representing the names of all the devices

GetDeviceStatus

Retrieves the status of a device

```
GetDeviceStatus(ByVal szName As String, ByRef pOpen As Integer, ByRef  
pCommsFailed As Integer, ByRef pInError As Integer)
```

Parameters	Description
szName	Name of the device
pStatus	Receives the status of the device

GetAppId

Returns a App ID string that can be used to identify the instance of this runtime.

```
GetAppId(ByRef pAppId As Object)
```

Parameters	Description
pAppId	Receives the App ID

IsValidAppId

This method determines whether the given App ID matches the current's instance's App ID.

```
IsValidAppId(ByVal szAppId As String, ByRef retVal As Integer)
```

Parameters	Description
szAppId	The App ID to validate
retVal	Receives true or false

IsValidUser

This method determines whether the given username and password matches a configured CX-Supervisor runtime user with web access enabled.

```
IsValidAppId(ByVal szUsername As String, ByVal szPassword As String,  
ByRef retVal As Integer)
```

Parameters	Description
szUsername	A username to validate
szPassword	A password to validate
retVal	Receives true or false

AlarmMngt

Allows the client to view alarm state and history and acknowledge active alarms.

ListAlarmGroups

Retrieves a list of all the alarm groups contained within a CX-Supervisor application.

```
ListAlarmGroups(ByRef pGroups As Object)
```

Parameters	Description
pGroups	Receives an array of strings representing the names of all the alarm groups

ListAlarms

Retrieves a list of all the alarm contained within the given alarm group.

```
ListAlarms(ByVal szName As String, ByRef pAlarms As Object)
```

Parameters	Description
szName	Name of Alarm Group
pAlarms	Receives an array of strings representing the names of all the alarms

GetAlarmData

Retrieves the metadata of a specific alarm.

```
GetAlarmData(ByVal szName As String, ByRef pType As Object, ByRef pAuto As Integer, ByRef pDescription As Object, ByRef pPriority As Object, ByRef pStatus As Object, ByRef pDateTime As Object, ByRef pMessage As Object)
```

Parameters	Description
szName	Name of the Alarm
pType	The type of Alarm "simple", "deadband", "rateofchange"
pAuto	Indicates Automatically acknowledged
pDescription	The description of the alarm
pPriority	The priority of the alarm "Highest", "High", "Medium", "Low", "Lowest"
pStatus	The current status of the alarm "Inactive", "Active", "Acknowledged"

pDateTime	The time and date the alarm entered it's current state
pMessage	The message shown

AcknowledgeAlarm

Acknowledge an alarm.

```
AcknowledgeAlarm(ByVal szName As String, ByVal szUser As String)
```

Parameters	Description
szName	Name of Alarm
szUser	User who acknowledged alarm

AcknowledgeAllAlarms

Acknowledge all un-acknowledged, active alarms.

```
AcknowledgeAllAlarms(ByVal szUser As String)
```

Parameters	Description
szName	Name of Alarm
szUser	User who acknowledged alarms

BrowseAlarms

This method retrieves a filtered list of the alarms contained within a CX-Supervisor application.

```
BrowseAlarms(ByVal szFilter As String, ByVal szPriorityFilter As String, ByRef pAlarms As Object)
```

Parameters	Description
szFilter	Free format filter eg. A*
szPriorityFilter	Alarm priority filter. Eg. "High". If string empty all types are returned
pAlarms	Receives an array of strings representing the names of all the alarms

GetAlarmLog

This method provides the ability to get a list of all the alarm log entries. The returned array strings each delimited by tabs will provide time, message and status information. The list matches the order of entries in the log.

```
GetAlarmLog(ByRef pAlarmLogEntries As Object)
```

Parameters	Description
pAlarmLogEntries	Receives an array of strings representing the alarm log entries

GetActiveAlarms

Retrieves a list of all the currently active alarms.

`GetActiveAlarms (ByRef pAlarms As Object)`

Parameters	Description
pAlarms	Receives an array of strings representing the names of all the alarms

ScriptMngt

Allows the client to execute scripts contained within a supervisor application.

ExecuteScript

Execute a project level script in the supervisor application.

```
ExecuteScript(ByVal varName As Object, ByRef pParamList As Object,  
ByRef retVal As Object)
```

Parameters	Description
varName	Name of script
pParamList	Array of objects representing the list of script arguments
retVal	Returned output parameters

ListScripts

Retrieve a list of all the project level scripts contained within a CX-Supervisor application.

```
ListScripts(ByRef pScripts As Object)
```

Parameters	Description
pScripts	Receives an array of strings representing the names of all the scripts

GetScriptParameters

Retrieves the parameters associated with a script.

```
GetScriptParameters(ByVal szName As String, ByRef pParamList As  
Object)
```

Parameters	Description
szName	The name of the script
pParamList	Receives an array of strings representing the data types of the parameters

ErrorMngt

Allows the client to access to the error log.

GetErrorLog

Get a list of all the error log entries.

`GetErrorLog(ByRef pErrors As Object)`

Parameters	Description
pErrors	Receives an array of strings representing all of the error log entries (date/time and message)