Cat No. W15E-EN-10

Software

CX-Supervisor

.NET Interface Reference

OMRON

Notice

OMRON products are manufactured for use by a trained operator and only for the purposes described in this manual.

The following conventions are used to classify and explain the precautions in this manual. Always heed the information provided with them.

Note: Indicates information of particular interest for efficient and convenient operation of the product.

Caution:
Indicates information that, if not heeded, could possibly result in minor or relatively serious injury, damage to the product, or faulty operation.

Warning:
Indicates information that, if not heeded, could possibly result in serious injury or loss of life.

Trademarks and copyrights

CX-Supervisor is a registered trademark of OMRON.

All other product names, company names, logos or other designations mentioned herein are trademarks of their respective owners.

Copyright

SECTION 1 Introduction

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

1-1 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.

1-2 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.

1-3 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.

SCSRunlib.dll is a .NET interop assembly. For maximum backwards compatibility it has been compiled as .NET Framework V2 which can be referenced by projects targeting later versions.

Note: If your platform does not recognise .NET Framework V2 references (for example Python 3) then contact your support team to request a compatible interop assembly.

SECTION 2
Examples

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

2-1 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…
```

2-2 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…
```

2-3 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.

SECTION 3
Class Reference

3-1 PointMngt

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

3-1-1 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

3-1-2 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

3-1-3 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

3-1-4 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

3-1-5 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

3-1-6 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

3-1-7 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

3-1-8 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

Parameters	Description
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

3-2 ApplicationMngt

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

3-2-1 GetErrorString

This method returns the error string corresponding to a specific error code generated by CXSupervisor 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

3-2-2 Restart

Causes the CX-Supervisor runtime to be restarted. If a path to a .SR3 file is specified then this is run upon restart.

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

Parameters	Description
szFile	Optional path of the CX-Supervisor application to run.

3-2-3 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

3-2-4 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

3-2-5 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

3-2-6 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

3-2-7 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

3-2-8 IsValidUser

This method determines whether the given username and password matches a configured CXSupervisor 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

3-3 AlarmMngt

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

3-3-1 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

3-3-2 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

3-3-3 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

3-3-4 AcknowledgeAlarm

Acknowledge an alarm.

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

Parameters	Description
szName	Name of Alarm
szUser	User who acknowledged alarm

3-3-5 AcknowledgeAllAlarms

Acknowledge all un-acknowledged, active alarms.

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

Parameters	Description
szName	Name of Alarm
szUser	User who acknowledged alarms

3-3-6 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

3-3-7 GetAlarmLog

This method provides the ability to get a list of all the alarm log entries. The returned array strings each delimitated 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

3-3-8 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

3-4 ScriptMngt

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

3-4-1 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

3-4-2 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

3-4-3 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

3-5 ErrorMngt

Allows the client to access to the error log.

3-5-1 GetErrorLog

Get a list of all the error log entries.

```
GetErrorLog(ByRef pErrors As Object)
```

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

Revision history

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.

Cat. No. W15E-EN-10

The following table lists the changes made to the manual during each revision. The page numbers of a revision refer to the previous version.

Revision code	Date	Revised content
01	Sept. 2010	First version in the standard Omron format.
02	June 2011	Updated for CX-Supervisor 3.2 release.
03	March 2017	Updated for CX-Supervisor 3.3 release.
04	Oct. 2017	Updated for CX-Supervisor 3.4 release.
05	Dec. 2018	Updated for CX-Supervisor 3.5 release.
06	Feb. 2020	Updated for CX-Supervisor 4.0 release.
07	June 2021	Updated for CX-Supervisor 4.1 release.
08	Jan. 2022	Updated for CX-Supervisor 4.2 release.
09	March 2023	Updated for CX-Supervisor 4.3 release.
10	April 2024	Updated for CX-Supervisor 4.4 release.

OMRON

Authorized Distributor:

Cat. No. W15E-EN-10

Note: Specifications subject to change without notice.

Printed in Europe