# Crestron **SDK for Microsoft® .NET Component**

Software

Introduction and Tutorial

CRESTRON®

This document was prepared and written by the Technical Documentation department at:

Crestron Electronics, Inc.
15 Volvo Drive
Rockleigh, NJ 07647
1-888-CRESTRON

# Contents

# Software: SDK for Microsoft® .NET Component

## Introduction

**NOTE:** This document assumes that the user has a proficient working knowledge of Microsoft Visual Studio, the .NET framework and associated languages, and is familiar with Crestron® control systems and their functions.

The Crestron Software Development Kit (SDK) for Microsoft® .NET Component is designed to allow developers to create .NET projects capable of bi-directional communication between a computer running the .NET component and Crestron 2-Series and X-Gen control systems.

This communication, employing Crestron's implementation of joins and signal types, allows for the creation of powerful dynamic interfaces that will run on systems running the .NET framework.

The .NET component is an Ethernet Intersystem/Device Communication symbol in SIMPL Windows that can be thought of as a touchpanel in that it can send and receive the three basic data types: *digital*, *analog*, and *serial*. These data types may take a variety of forms corresponding to the various controls available in the SDK, but the control events sent to the control system would take some form of:

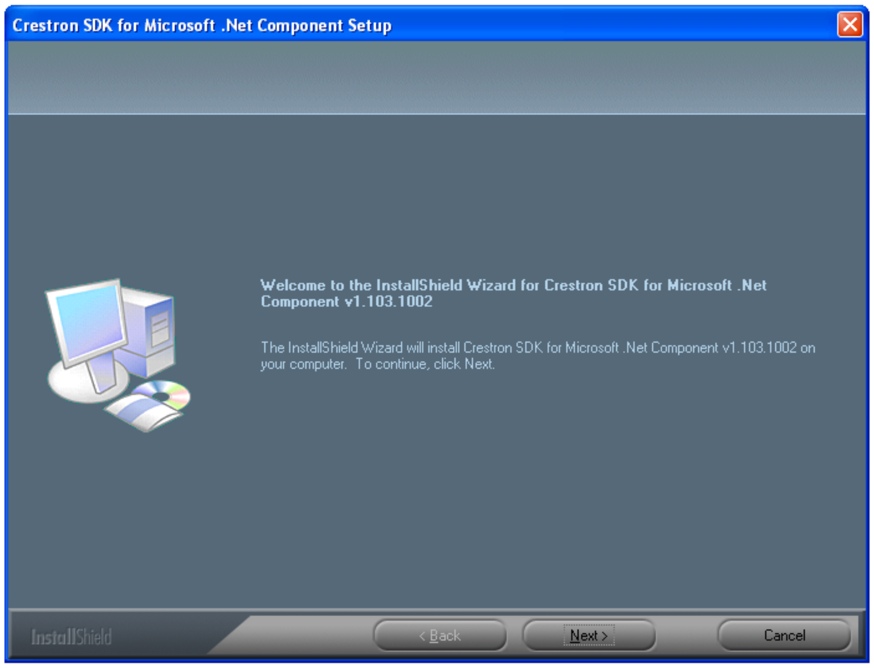
- onDigital
- onAnalog
- onSerial

## Installing the SDK for Microsoft .NET Component

### Automatic Installation

Download the SDK for Microsoft .NET Component installer from the Crestron website to your desktop. Double click the file to automatically start the installation wizard. The Crestron SDK for Microsoft .NET Component "Setup Welcome" window appears.

*Crestron SDK for Microsoft .NET Component "Setup Welcome" Window*

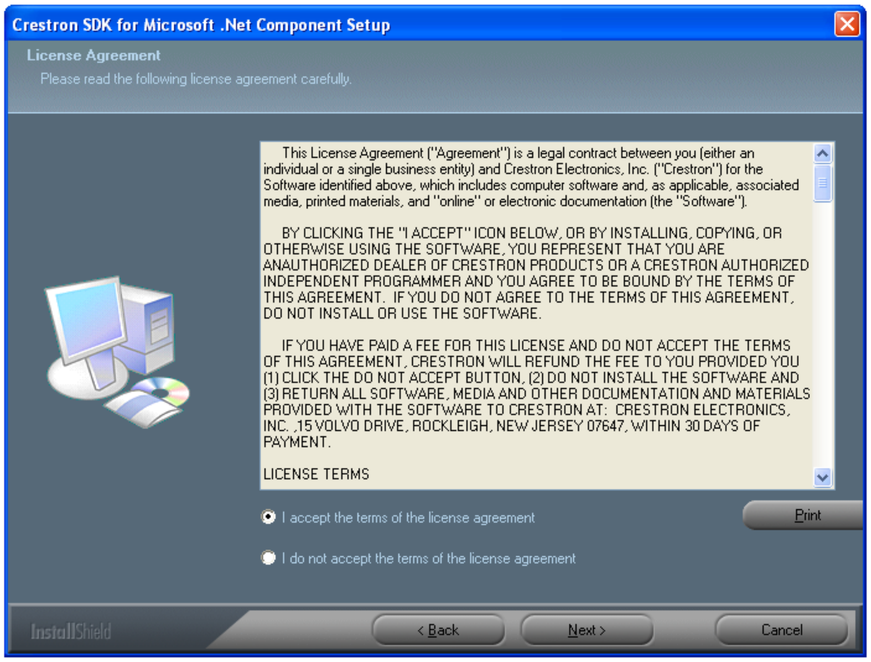


**Welcome to the InstallShield Wizard for Crestron SDK for Microsoft .Net Component v1.103.1002**

The InstallShield Wizard will install Crestron SDK for Microsoft .Net Component v1.103.1002 on your computer. To continue, click Next.

1. Click **Next** to display the "License Agreement" window where you will be prompted to accept (or not) the license agreement governing the .NET component. Accept the agreement (if you want to continue) by clicking the appropriate radio button. You can also use the **Print** button to retain a copy for your records. Click **Next** to continue.

*"License Agreement" Window*



Crestron SDK for Microsoft .Net Component Setup
License Agreement
Please read the following license agreement carefully.
Print
InstallShield
< Back
Next >
Cancel

This License Agreement ("Agreement") is a legal contract between you (either an individual or a single business entity) and Crestron Electronics, Inc. ("Crestron") for the Software identified above, which includes computer software and, as applicable, associated media, printed materials, and "online" or electronic documentation (the "Software").

BY CLICKING THE "I ACCEPT" ICON BELOW, OR BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE, YOU REPRESENT THAT YOU ARE ANAUTHORIZED DEALER OF CRESTRON PRODUCTS OR A CRESTRON AUTHORIZED INDEPENDENT PROGRAMMER AND YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT INSTALL OR USE THE SOFTWARE.

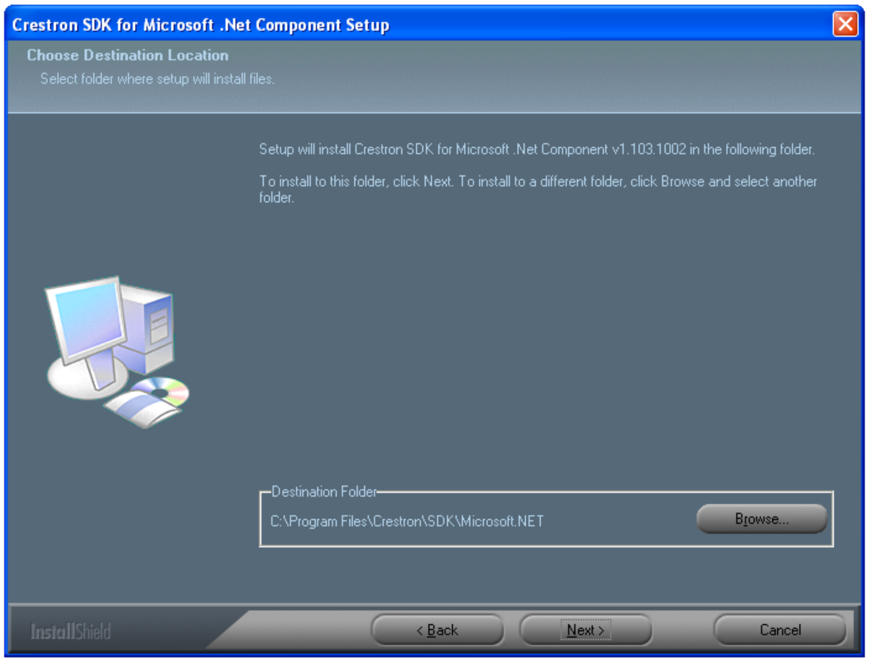
IF YOU HAVE PAID A FEE FOR THIS LICENSE AND DO NOT ACCEPT THE TERMS OF THIS AGREEMENT, CRESTRON WILL REFUND THE FEE TO YOU PROVIDED YOU (1) CLICK THE DO NOT ACCEPT BUTTON, (2) DO NOT INSTALL THE SOFTWARE AND (3) RETURN ALL SOFTWARE, MEDIA AND OTHER DOCUMENTATION AND MATERIALS PROVIDED WITH THE SOFTWARE TO CRESTRON AT: CRESTRON ELECTRONICS, INC. ,15 VOLVO DRIVE, ROCKLEIGH, NEW JERSEY 07647, WITHIN 30 DAYS OF PAYMENT.

LICENSE TERMS

[✓ I accept the terms of the license agreement]
I do not accept the terms of the license agreement

2. The “Choose Destination” window allows you to select the destination of the install. It is recommended that you accept the default unless your installation has other requirements.

***“Choose Destination” Window***


Crestron SDK for Microsoft .Net Component Setup

Choose Destination Location

Select folder where setup will install files.

Setup will install Crestron SDK for Microsoft .Net Component v1.103.1002 in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder
C:\Program Files\Crestron\SDK\Microsoft.NET

3. Click **Next** to complete the installation.

## Getting Started

1. Start Microsoft Visual Studio, select **Open Project** from the **File** menu and browse to the file: **SDK for Microsoft .NET Component Test Program.sln** or **SDK for Microsoft .NET Component Test Program.csproj** and open it. The ActiveCNX SDK for .NET example project opens.

# ActiveCNX.NET SDK Component Reference

The methods and events are all part of the class known as ActiveCNX.

**NOTE:** There may be additional parameters, events and return types for any given method. Refer to the Microsoft Visual Studio help file for more information.

### *Class ActiveCNX*

**Package:** Crestron
**Class:** ActiveCNX
**Member of:** Crestron.ActiveCNX
**Language:** C# for .NET

The ActiveCNX class handles the communication between a computer running the ActiveCNX control and Crestron 2-Series and X-Gen control systems.

*Constructor Detail*

#### ActiveCNX()

```
public ActiveCNX()
```

This is the default constructor.

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

#### ActiveCNX(string, long, string, string, int, bool)

```
public ActiveCNX(string strIPAddress, long lIPID, string
strUserName, string strPassword, int intPort, bool
boolSSL)
```

This constructor initializes the object, and connects it to the control system corresponding to the listed constructor arguments.

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

#### ActiveCNX(string, long, string, string, int, bool, int, int)

```
public ActiveCNX(string strIPAddress, long lIPID, string
strUserName, string strPassword, int intPort, bool
boolSSL, int iHeartBeatInterval, int iHeartBeatTimeOut)
```

This constructor initializes the object, and connects it to the control system corresponding to the listed constructor arguments.

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

*Method Details*

### Connect()

`public void Connect()`

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

This is the default connect method. If you have set the connection related properties of this object (either via the constructor, or direct assignment,) this version can be used. If the properties are not defined, one of the other connect methods must be used.

**Return Type:**

`void`

**Events:**

`onConnect` — Triggered when a connection is established with the control system.

`onError` — Triggered when an error is generated by the connection.

### Connect(string, long)

`public void Connect(string` *`strIPAddress`*`, long` *`lIPID`*`)`

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

This connect method is used when the string and long properties are not defined in the ActiveCNX class, such as when ActiveCNX() is used.

**Arguments:**

`string strIPAddress` — The IP Address of the control system expressed as a string.

`long lIPID` — The Crestron Internet Protocol ID (IP ID) expressed as a string. The default is 03.

**Return Type:**

`void`

**Events:**

`onConnect` — Triggered when a connection is established with the control system.

`onError` — Triggered when an error is generated by the connection.

### Connect(string, long, bool)

`public void Connect(string` *`strConnectTo`*`, long` *`lIPID`*`,`
`bool` *`boolSSL`*`)`

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

This connect method is used when the string, long and Boolean properties are not defined in the ActiveCNX class, such as when ActiveCNX() is used.

**Arguments:**

`string strIPAddress` — The IP Address of the control system expressed as a string.

`long lIPID` — The Crestron Internet Protocol ID (IPID) expressed as a string. The default is 03.

`bool boolSSL` — Status of SSL expressed as a Boolean.

**Return Type:**

`void`

**Events:**

`onConnect` — Triggered when a connection is established with the control system.

`onError` — Triggered when an error is generated by the connection.

`onCertVerification` — Triggered when the SSL is invoked. Returns SSL certification information as a string.

### Connect(string, long, string, string, int, bool, int, int)

```
public void Connect(string strConnectTo, long lIPID,
string strUserName, string strPassword int iPort, bool
boolSSL, int iHeartBeatInterval, int iHeartBeatTimeOut)
```

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

This connect method is used when the string, long, Boolean and int arguments are not defined in the ActiveCNX class, such as when ActiveCNX() is used.

**Arguments:**

`string strIPAddress` — The IP Address of the control system expressed as a string.

`long lIPID` — The Crestron Internet Protocol ID (IPID) expressed as a string. The default is 03.

`string strUserName` — The network username expressed as a string.

`string strPassword` — The network password expressed as a string.

`int iPort` — The IP port expressed as an integer.

`bool boolSSL` — Status of SSL expressed as a Boolean.

`int iHeartBeatInterval` — Sets the number of milliseconds between heartbeats sent to the control system.

`int iHeartBeatTimeOut` — The number of milliseconds where no heartbeat response is received before causing an "onError" event. The displayed message will read: "Heartbeat timeout occurred".

**NOTE:** The iHeartBeatTimeOut number must be larger than the iHeartBeatInterval.

**Return Type:**

`void`

**Events:**

`onConnect` — Triggered when a connection is established with the control system.

`onError` — Triggered when an error is generated by the connection.

`onCertVerification` — Triggered when the SSL is invoked. Returns SSL certification information as a string.

### Disconnect()

```
public void Disconnect()
```

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The Disconnect method is used to monitor the connection to the control system.

**Return Type:**

`void`

**Events:**

`onDisconnect` — Triggered when the connection to the control system has been lost.

`onError` — Triggered when an error is generated by the connection.

### SendAnalog(int, int)

```
public bool SendAnalog(int iJoin, int iValue)
```

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to pass join and value variables to the control system when the **Send** button is clicked and **Analog** is selected as the signal type.

**Arguments:**

`int iJoin` — The join number sent to the control system expressed as an integer.

`int iValue` — The contents of the Value field sent to the control system and expressed as an integer.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onAnalog` — Triggered when SendAnalog() is initiated.

`onError` — Triggered when the analog value fails to send.

### SendAnalog(int, int, int)

`public bool SendAnalog(int iSlot, int iJoin, int iValue)`

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to pass slot, join and value variables to the control system when the **Send** button is clicked and **Analog** is selected as the signal type.

**Arguments:**

`int iSlot` — The slot number selected from the Slot combo box specifies the slot for a sub-slotted symbol.

`int iJoin` — The join number sent to the control system expressed as an integer.

`int iValue` — The contents of the *Value* field sent to the control system and expressed as an integer.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onAnalog` — Triggered when SendAnalog() is initiated.

`onError` — Triggered when the analog value fails to send.

### SendDigital(int, bool)

`public bool SendDigital(int iJoin, bool boolValue)`

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to pass join and boolean variables to the control system when the **Send** button is clicked and **Digital** is selected as the signal type.

**Arguments:**

`int iJoin` — The join number sent to the control system, expressed as an integer.

`bool boolvalue` — True or false sent to the control system, expressed as 1 or 2.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onDigital` — Triggered when SendDigital() is initiated.

`onError` — Triggered when sending a digital fails.

### SendDigital(int, int, bool)

public bool SendDigital(int *iSlot*, int *iJoin*, bool *boolValue*)

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to pass slot, join and boolean variables to the control system when the **Send** button is clicked and **Digital** is selected as the signal type.

**Argument:**

`int iSlot` — The slot number selected from the Slot combo box specifies the slot for a sub-slotted symbol.

`int iJoin` — The join number sent to the control system, expressed as an integer.

`bool boolvalue` — True or false sent to the control system, expressed as 1 or 2.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onDigital` — Triggered when SendDigital() is initiated.

`onError` — Triggered when sending a digital fails.

### StartRepeatingDigital(int)

public bool StartRepeatingDigital(int *iJoin*)

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to send a continuous “high” signal to compensate for poorer networks (e.g., wireless networks) to ensure that the digital is dropped before a TCP Disconnect occurs.

**Argument:**

`int iJoin` — The join number sent to the control system, expressed as an integer.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onDigital` — Triggered when StartRepeatingDigital() is initiated.

`onError` — Triggered if the repeating digital fails to start.

### StartRepeatingDigital(int, int)

```
public bool StartRepeatingDigital(int iSlot, int iJoin)
```

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to send a continuous "high" signal to compensate for poorer networks (e.g., wireless networks) to ensure that the digital is dropped before a TCP Disconnect occurs.

**Argument:**

`int iSlot` — The slot number selected from the Slot combo box specifies the slot for a sub-slotted symbol.

`int iJoin` — The join number sent to the control system, expressed as an integer.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onDigital` — Triggered when StartRepeatingDigital() is initiated.

`onError` — Triggered if the repeating digital fails to start.

### StopRepeatingDigital(int)

```
public bool StopRepeatingDigital(int iJoin)
```

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to stop the repeating digital initiated by StartRepeatingDigital().

**Argument:**

`int iJoin` — The join number sent to the control system, expressed as an integer.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onDigital` — Triggered when StopRepeatingDigital() is initiated.

`onError` — Triggered if it fails to stop the repeating digital.

### StopRepeatingDigital(int, int)

`public bool StopRepeatingDigital(int iSlot, int iJoin)`

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to stop the repeating digital initiated by StartRepeatingDigital().

**Argument:**

`int iSlot` — The slot number selected from the Slot combo box specifies the slot for a sub-slotted symbol.

`int iJoin` — The join number sent to the control system, expressed as an integer.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onDigital` — Triggered when StopRepeatingDigital() is initiated.

`onError` — Triggered if it fails to stop the repeating digital.

### SendSerial(int, string)

`public bool SendSerial(int iJoin, string strValue)`

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to pass join and string variables to the control system when the **Send** button is clicked and **Serial** is selected as the signal type.

**Arguments:**

`int iJoin` — The join number sent to the control system, expressed as an integer.

`string strValue` — A string passed to the control system.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onSerial` — Triggered when SendSerial() is initiated.

`onError` — Triggered if SendSerial fails.

### SendSerial(int, int, string)

```
public bool SendSerial(int iSlot, int iJoin, string
strValue)
```

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The method used to pass slot, join and string variables to the control system when the **Send** button is clicked and **Serial** is selected as the signal type.

**Arguments:**

`int iSlot` — The slot number selected from the Slot combo box specifies the slot for a sub-slotted symbol.

`int iJoin` — The join number sent to the control system, expressed as an integer.

`string strValue` — A string passed to the control system.

**Return Type:**

`boolean` — True equals Success. False equals Fail.

**Events:**

`onSerial` — Triggered when SendSerial() is initiated.

`onError` — Triggered if SendSerial fails.

### UpdateRequest()

```
public void UpdateRequest()
```

**Member of:** Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

The UpdateRequest method requests the status of all attributes assigned to the symbol, except for attributes with a value of zero or false.

**NOTE:** Defined member variables tied to attribute values should be zeroed out to avoid invalid values.

**Return Type**

`void`

**Events:**

`onConnect` — Triggered when a connection is established with the control system.

`onError` — Triggered when an error is generated by the connection.

*Event Details*

### onAnalog

```
public event Crestron.ActiveCNX.ActiveCNXEventHandler
onAnalog
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Triggered when an analog join is received from the control system.

| PROPERTY | VALUE |
|---|---|
| Slot | Integer—The slot for a sub-slotted symbol. |
| Join | Integer—The join number. |
| AnalogValue | Integer—The analog value (0 to 65535) sent from the control system. |

### onCertVerification

```
public event
Crestron.ActiveCNX.ActiveCNXCertificateEventHandler
onCertVerification
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Triggered when SSL certificate/verification information is received from the control system.

| PROPERTY | VALUE |
|---|---|
| Certificate | String—The SSL certificate. |
| VerificationResults | String—The results of verifying the SSL certificate. |

### onConnect

```
public event
Crestron.ActiveCNX.ActiveCNXConnectionEventHandler
onConnect
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Triggered when a connection is made with a control system.

### onDigital

```
public event Crestron.ActiveCNX.ActiveCNXEventHandler
onDigital
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Triggered when a digital join is received from the control system.

| PROPERTY | VALUE |
|---|---|
| Slot | Integer—The slot for a sub-slotted symbol. |
| Join | Integer—The join number. |
| DigitalValue | Boolean—The digital value (True/False) sent from the control system. |

### onDisconnect

```
public event
Crestron.ActiveCNX.ActiveCNXConnectionEventHandler
onDisconnect
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Triggered when the connection to a control system is broken.

| PROPERTY | VALUE |
|---|---|
| DisconnectReasonMessage | String—Message sent by control system indicating the reason for the disconnect. |

### onError

```
public event
Crestron.ActiveCNX.ActiveCNXErrorEventHandler onError
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Triggered when an error is generated by ActiveCNX.

| PROPERTY | VALUE |
|---|---|
| ErrorMessage | String—The error message received from the control system. |

### onSerial

```
public event Crestron.ActiveCNX.ActiveCNXEventHandler
onSerial
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Triggered when a serial join is received from the control system.

| PROPERTY | VALUE |
|---|---|
| Slot | Integer—The slot for a sub-slotted symbol. |
| Join | Integer—The join number. |
| SerialValue | String—The serial value sent from the control system. |

## *Properties Details*

### IPAddress

```
public string IPAddress { get; }
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Used to read and return the IP address of the ActiveCNX object.

### IPID

```
public long IPID { get; }
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Used to read and return the IP ID of the control system.

### IPPort

```
public long IPPort { get; }
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Used to read and return the IP port of the control system.

### isConnected

```
public bool isConnected { get; }
```

**Member of**: Crestron.ActiveCNX.ActiveCNX
**Language Version:** C# for .NET

Reads and returns the connection status of the control system.

*EventArg Classes*

### ActiveCNXEventArgs()

```
public class ActiveCNXEventArgs : System.EventArgs
```

**Member of:** Crestron.ActiveCNX
**Language Version:** C# for .NET

This is the base ActiveCNXEventArg class.

### ActiveCNXEventArgs(int, int, bool)

```
public ActiveCNXEventArgs(int iSlot, int iJoin, bool
boolDigitalValue)
```

**Member of:** Crestron.ActiveCNX.ActiveCNXEventArgs
**Language Version:** C# for .NET

This initializes the properties corresponding to the listed arguments.

### ActiveCNXEventArgs(int, int, int)

```
public ActiveCNXEventArgs(int iSlot, int iJoin, int
iAnalogValue)
```

**Member of:** Crestron.ActiveCNX.ActiveCNXEventArgs
**Language Version:** C# for .NET

Initializes the properties corresponding to the listed arguments.

### ActiveCNXEventArgs(int, int, string)

```
public ActiveCNXEventArgs(int iSlot, int iJoin, string
strSerialValue)
```

**Member of:** Crestron.ActiveCNX.ActiveCNXEventArgs
**Language Version:** C# for .NET

Initializes the properties corresponding to the listed arguments.

### ActiveCNXErrorEventArgs()

```
public class ActiveCNXErrorEventArgs : System.EventArgs
```

**Member of:** Crestron.ActiveCNX
**Language Version:** C# for .NET

This is the base ActiveCNXErrorEventArg class.

### ActiveCNXErrorEventArgs(int, string)

```
public ActiveCNXErrorEventArgs(int iNumber, string
strMessage)
```

**Member of:** Crestron.ActiveCNX.ActiveCNXErrorEventArgs
**Language Version:** C# for .NET

Initializes the properties corresponding to the listed arguments.

### ActiveCNXConnectionEventArgs()

```
public class ActiveCNXConnectionEventArgs :
System.EventArgs
```

**Member of:** Crestron.ActiveCNX
**Language Version:** C# for .NET

This is the base ActiveCNXConnectionEventArgs class.

### ActiveCNXConnectionEventArgs(bool, int, string)

```
public ActiveCNXConnectionEventArgs(bool boolConnected,
int iParam, string strParam)
```

**Member of:** Crestron.ActiveCNX.ActiveCNXConnectionEventArgs
**Language Version:** C# for .NET

Initializes the properties corresponding to the listed arguments.

### ActiveCNXCertificateEventArgs()

```
public class ActiveCNXCertificateEventArgs :
System.EventArgs
```

**Member of:** Crestron.ActiveCNX
**Language Version:** C# for .NET

This is the base ActiveCNXCertificateEventArgs class.

### ActiveCNXCertificateEventArgs(string, string)

```
public ActiveCNXCertificateEventArgs(string
strCertificate, string strCertVerification)
```

**Member of:** Crestron.ActiveCNX.ActiveCNXCertificateEventArgs
**Language Version:** C# for .NET

Initializes the properties corresponding to the listed arguments.

# How to Use the Code

The following examples show how the code can be used to initialize the ActiveCNX component to connect to a device, handle join events and show connection issues. These examples are taken from the example project opened in "Getting Started" on page 3 of this manual. They are representative only and may not apply to your situation.

## Initialization

1. Establish your base classes (or libraries or namespaces) with the using directive.

**NOTE:** It is only necessary to establish namespaces needed for your project.

```
using System;
using System.Diagnostics;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Threading;
using System.Windows.Forms;
using Data;
```

2. Establish the Crestron namespace.

```
using Crestron.ActiveCNX;
```

3. Establish the namespace of the example program.

```
namespace ActiveCNX_Test_Program
```

4. Initialize Form1 as required for Windows Form Designer support. This block of code populates the IP ID combo box with all possible IP IDs, creates the ActiveCNX() class object in memory, allows event handlers to update the *Received from Control System* text box, as well as defines event handlers.

***Example Program Code***

```
public Form1()
{
       InitializeComponent();

      this.ddlLogging.SelectedIndex = 0;
       for (int x = 3; x < 255; x++)
       {
       this.cbIPID.Items.Add(string.Format("{0:X2}",x));
       }
       this.cbIPID.SelectedIndex = 0;
       this.cbValue.SelectedIndex = 0;
       this.cbSlot.SelectedIndex = 0;
       this.acnxConnection = new ActiveCNX();
       this.cbValue.Hide();
       this.tbValue.Hide();
       this.lblValue.Hide();

       this.acnxConnection.onConnect += new
Crestron.ActiveCNX.ActiveCNXConnectionEventHandler(acnxConnection_onConnect);
       this.acnxConnection.onDisconnect +=new
Crestron.ActiveCNX.ActiveCNXConnectionEventHandler(acnxConnection_onDisconnect);
       this.acnxConnection.onError += new
Crestron.ActiveCNX.ActiveCNXErrorEventHandler(acnxConnection_onError);
       this.acnxConnection.onAnalog +=new
Crestron.ActiveCNX.ActiveCNXEventHandler(acnxConnection_onAnalog);
       this.acnxConnection.onDigital += new
Crestron.ActiveCNX.ActiveCNXEventHandler(acnxConnection_onDigital);
       this.acnxConnection.onSerial += new
Crestron.ActiveCNX.ActiveCNXEventHandler(acnxConnection_onSerial);
       this.acnxConnection.onCertVerification += new
Crestron.ActiveCNX.ActiveCNXCertificateEventHandler(acnxConnection_onCertVerificati
on);

        Control.CheckForIllegalCrossThreadCalls = false;
}
```

## Connecting to a Control System

All connection and interaction with a control system is accomplished via the buttons, selections and text entries on a compiled interface such as the example program. The actions they generate are controlled by event handlers.

In order to make a connection to a control system the IP address, IP ID and IP port must be declared by selections on the interface. Once this is done, the **Connect** button can be clicked which will attempt the connection.

**NOTE:** Adding an event handler for a acnxConnection_onDisconnect at the same time helps monitor the connection.

1. The btnConnect_Click handler will establish the IP address and IP ID as well as validate the user name, password, IP port and whether SSL is being used.
2. When the connection is established, the acnxConnection_onConnect handler is triggered. This prints update request information to the *Received from Control System* field and flips the states of the **Connect** and **Disconnect** buttons appropriately.

3. Implement the acnxConnection_onDisconnect handler as one way of monitoring the connection.

**NOTE:** The acnxConnection_onError handler can also be implemented to monitor error events generated by ActiveCNX.

## Handling Join Events

The acnxConnection_onAnalog, acnxConnection_onDigital, acnxConnection_onSerial and btnSend_Click event handlers enable the ActiveCNX SDK for .NET to both send and receive join information.

The acnxConnection_onAnalog, acnxConnection_onDigital and acnxConnection_onSerial event handlers are triggered when an analog, digital or serial join is received from the control system. The btnSend_Click event handler is triggered when the **Send** button on the interface is clicked and transmits the entered join, slot and value information.

1. Add the event handlers to handle join information sent from the control system.
2. Add the event handler to send join, slot and value information to the control system.

## Showing Connection Problems

To monitor the connection, enable the acnxConnection_onDisconnect and acnxConnection_onError event handlers.

1. Add the acnxConnection_onDisconnect event handler to listen for a disconnect from the control system.
2. Add the acnxConnection_onError event handler to monitor error events generated by ActiveCNX that indicate a connection problem.

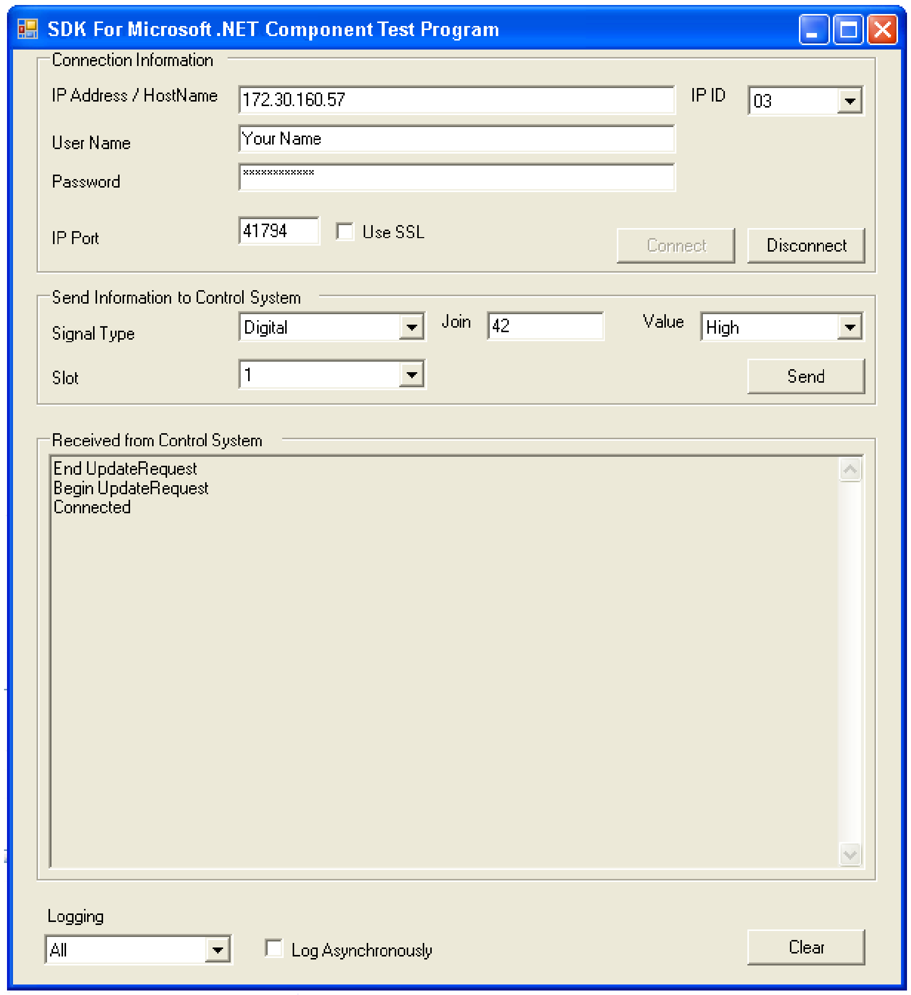
## The Example Program

The ActiveCNX for .NET Example Program is available from the Crestron website and shows one possible way to use the ActiveCNX for .NET SDK to create an interface to a 2-Series or X-Gen control system.

Opening the example program in Microsoft Visual Studio will provide you with insight regarding the creation and implementation of a C# .NET project suitable for your needs. Running the program illustrates the various features you will find in the code.

1. Open the project file: **SDK for Microsoft .NET Component Test Program.sln** or **SDK for Microsoft .NET Component Test Program.csproj** in Visual Studio and press **F5** to compile and run in Debug mode. The example program, Form 1, shown below, opens.

***Crestron ActiveCNX for .NET Example Program***



IP Address / HostName: 172.30.160.57
IP ID: 03
User Name: Your Name
Password: ************
IP Port: 41794

Signal Type: Digital
Join: 42
Value: High
Slot: 1

```
End UpdateRequest
Begin UpdateRequest
Connected
```

Logging: All

2. Enter an IP address or host name in the *IP Address/HostName* text box, select the appropriate IP ID from the IP ID combo box, enter the appropriate IP port in the *IP Port* text box. If required by your control

system, enter a username and password in the text boxes by the same names and click **Connect**.

3. When the connection is made, the **Connect** button is grayed out and the **Disconnect** button takes focus. Any messages that result are displayed in the *Received from Control System* field.
4. To send a signal to the control system, choose the type of signal from the Signal Type combo box, enter the appropriate join number in the *Join* field, select or enter the appropriate signal in the *Value* field and click **Send**.

**NOTE:** If you are monitoring the control system and its installed symbol(s) via the Toolbox Text Consol or the SIMPL debugger, you will see the results of the values you send.

5. Errors and control system responses to events are displayed in the *Received from Control System* field.

## Software License Agreement

This License Agreement ("Agreement") is a legal contract between you (either an individual or a single business entity) and Crestron Electronics, Inc. ("Crestron") for software referenced in this guide, which includes computer software and as applicable, associated media, printed materials and "online" or electronic documentation (the "Software").

BY INSTALLING, COPYING OR OTHERWISE USING THE SOFTWARE, YOU REPRESENT THAT YOU ARE AN AUTHORIZED DEALER OF CRESTRON PRODUCTS OR A CRESTRON AUTHORIZED INDEPENDENT PROGRAMMER AND YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT INSTALL OR USE THE SOFTWARE.

IF YOU HAVE PAID A FEE FOR THIS LICENSE AND DO NOT ACCEPT THE TERMS OF THIS AGREEMENT, CRESTRON WILL REFUND THE FEE TO YOU PROVIDED YOU (1) CLICK THE DO NOT ACCEPT BUTTON, (2) DO NOT INSTALL THE SOFTWARE AND (3) RETURN ALL SOFTWARE, MEDIA AND OTHER DOCUMENTATION AND MATERIALS PROVIDED WITH THE SOFTWARE TO CRESTRON AT: CRESTRON ELECTRONICS, INC., 15 VOLVO DRIVE, ROCKLEIGH, NEW JERSEY 07647, WITHIN 30 DAYS OF PAYMENT.

### LICENSE TERMS

Crestron hereby grants You and You accept a nonexclusive, nontransferable license to use the Software (a) in machine readable object code together with the related explanatory written materials provided by Crestron (b) on a central processing unit ("CPU") owned or leased or otherwise controlled exclusively by You and (c) only as authorized in this Agreement and the related explanatory files and written materials provided by Crestron.

If this software requires payment for a license, you may make one backup copy of the Software, provided Your backup copy is not installed or used on any CPU. You may not transfer the rights of this Agreement to a backup copy unless the installed copy of the Software is destroyed or otherwise inoperable and You transfer all rights in the Software.

You may not transfer the license granted pursuant to this Agreement or assign this Agreement without the express written consent of Crestron.

If this software requires payment for a license, the total number of CPU's on which all versions of the Software are installed may not exceed one per license fee (1) and no concurrent, server or network use of the Software (including any permitted back-up copies) is permitted, including but not limited to using the Software (a) either directly or through commands, data or instructions from or to another computer (b) for local, campus or wide area network, internet or web hosting services or (c) pursuant to any rental, sharing or "service bureau" arrangement.

The Software is designed as a software development and customization tool. As such Crestron cannot and does not guarantee any results of use of the Software or that the Software will operate error free and You acknowledge that any development that You perform using the Software or Host Application is done entirely at Your own risk.

The Software is licensed and not sold. Crestron retains ownership of the Software and all copies of the Software and reserves all rights not expressly granted in writing.

### OTHER LIMITATIONS

You must be an Authorized Dealer of Crestron products or a Crestron Authorized Independent Programmer to install or use the Software. If Your status as a Crestron Authorized Dealer or Crestron Authorized Independent Programmer is terminated, Your license is also terminated.

You may not rent, lease, lend, sublicense, distribute or otherwise transfer or assign any interest in or to the Software.

You may not reverse engineer, decompile or disassemble the Software.

You agree that the Software will not be shipped, transferred or exported into any country or used in any manner prohibited by the United States Export Administration Act or any other export laws, restrictions or regulations ("Export Laws"). By downloading or installing the Software You (a) are certifying that You are not a national of Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria or any country to which the United States embargoes goods (b) are certifying that You are not otherwise prohibited from receiving the Software and (c) You agree to comply with the Export Laws.

If any part of this Agreement is found void and unenforceable, it will not affect the validity of the balance of the Agreement, which shall remain valid and enforceable according to its terms. This Agreement may only be modified by a writing signed by an authorized officer of Crestron. Updates may be licensed to You by Crestron with additional or different terms. This is the entire agreement between Crestron and You relating to the Software and it supersedes any prior representations, discussions, undertakings, communications or advertising relating to the Software. The failure of either party to enforce any right or take any action in the event of a breach hereunder shall constitute a waiver unless expressly acknowledged and set forth in writing by the party alleged to have provided such waiver.

If You are a business or organization, You agree that upon request from Crestron or its authorized agent, You will within thirty (30) days fully document and certify that use of any and all Software at the time of the request is in conformity with Your valid licenses from Crestron of its authorized agent.

Without prejudice to any other rights, Crestron may terminate this Agreement immediately upon notice if you fail to comply with the terms and conditions of this Agreement. In such event, you must destroy all copies of the Software and all of its component parts.

PROPRIETARY RIGHTS

Copyright. All title and copyrights in and to the Software (including, without limitation, any images, photographs, animations, video, audio, music, text and "applets" incorporated into the Software), the accompanying media and printed materials and any copies of the Software are owned by Crestron or its suppliers. The Software is protected by copyright laws and international treaty provisions. Therefore, you must treat the Software like any other copyrighted material, subject to the provisions of this Agreement.

Submissions. Should you decide to transmit to Crestron's website by any means or by any media any materials or other information (including, without limitation, ideas, concepts or techniques for new or improved services and products), whether as information, feedback, data, questions, comments, suggestions or the like, you agree such submissions are unrestricted and shall be deemed non-confidential and you automatically grant Crestron and its assigns a non-exclusive, royalty-free, worldwide, perpetual, irrevocable license, with the right to sublicense, to use, copy, transmit, distribute, create derivative works of, display and perform the same.

Trademarks. CRESTRON and the Swirl Logo are registered trademarks of Crestron Electronics, Inc. You shall not remove or conceal any trademark or proprietary notice of Crestron from the Software including any back-up copy.

GOVERNING LAW

This Agreement shall be governed by the laws of the State of New Jersey, without regard to conflicts of laws principles. Any disputes between the parties to the Agreement shall be brought in the state courts in Bergen County, New Jersey or the federal courts located in the District of New Jersey. The United Nations Convention on Contracts for the International Sale of Goods shall not apply to this Agreement.

CRESTRON LIMITED WARRANTY

CRESTRON warrants that: (a) the Software will perform substantially in accordance with the published specifications for a period of ninety (90) days from the date of receipt and (b) that any hardware accompanying the Software will be subject to its own limited warranty as stated in its accompanying written material. Crestron shall, at its option, repair or replace or refund the license fee for any Software found defective by Crestron if notified by you within the warranty period. The foregoing remedy shall be your exclusive remedy for any claim or loss arising from the Software.

CRESTRON shall not be liable to honor warranty terms if the product has been used in any application other than that for which it was intended or if it as been subjected to misuse, accidental damage, modification or improper installation procedures. Furthermore, this warranty does not cover any product that has had the serial number or license code altered, defaced, improperly obtained or removed.

Notwithstanding any agreement to maintain or correct errors or defects, Crestron shall have no obligation to service or correct any error or defect that is not reproducible by Crestron or is deemed in Crestron's reasonable discretion to have resulted from (1) accident; unusual stress; neglect; misuse; failure of electric power, operation of the Software with other media not meeting or not maintained in accordance with the manufacturer's specifications or causes other than ordinary use; (2) improper installation by anyone other than Crestron or its authorized agents of the Software that deviates from any operating procedures established by Crestron in the material and files provided to You by Crestron or its authorized agent; (3) use of the Software on unauthorized hardware or (4) modification of, alteration of or additions to the Software undertaken by persons other than Crestron or Crestron's authorized agents.

ANY LIABILITY OF CRESTRON FOR A DEFECTIVE COPY OF THE SOFTWARE WILL BE LIMITED EXCLUSIVELY TO REPAIR OR REPLACEMENT OF YOUR COPY OF THE SOFTWARE WITH ANOTHER COPY OR REFUND OF THE INITIAL LICENSE FEE CRESTRON RECEIVED FROM YOU FOR THE DEFECTIVE COPY OF THE PRODUCT. THIS WARRANTY SHALL BE THE SOLE AND EXCLUSIVE REMEDY TO YOU. IN NO EVENT SHALL CRESTRON BE LIABLE FOR INCIDENTAL, CONSEQUENTIAL, SPECIAL OR PUNITIVE DAMAGES OF ANY KIND (PROPERTY OR ECONOMIC DAMAGES INCLUSIVE), EVEN IF A CRESTRON REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR OF ANY CLAIM BY ANY THIRD PARTY. CRESTRON MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO TITLE OR INFRINGEMENT OF THIRD-PARTY RIGHTS, MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY OTHER WARRANTIES, NOR AUTHORIZES ANY OTHER PARTY TO OFFER ANY WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY FOR THIS PRODUCT. THIS WARRANTY STATEMENT SUPERSEDES ALL PREVIOUS WARRANTIES.

This page is intentionally left blank.

**Crestron Electronics, Inc.**
15 Volvo Drive   Rockleigh, NJ 07647
Tel: 888.CRESTRON
Fax: 201.767.7576
www.crestron.com

**Introduction & Tutorial – DOC. 6718A**
**(2022313)**
**12.08**
Specifications subject to
change without notice.