

# **Integration Objects'**

## **OPC Connectivity Solution for StreamInsight**

**OPC Adapter for Microsoft StreamInsight**

Version 1.0 Rev.1

**USER GUIDE**

OPC Adapter for Microsoft StreamInsight Developer's User Guide Version 1.0 Rev .1  
Published September 2014

Copyright © 2012-2014 Integration objects. All rights reserved.

No part of this document may be reproduced, stored in a retrieval system, translated, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Integration Objects.

Windows<sup>®</sup>, Windows NT<sup>®</sup> and .NET are registered trademarks of Microsoft Corporation.

# TABLE OF CONTENTS

<b>PREFACE .....</b>	<b>7</b>
About This User Guide .....	7
Target Audience .....	7
Document Conventions .....	7
Customer Support Services .....	7
<b>INTRODUCTION .....</b>	<b>8</b>
1. Overview.....	8
2. Architecture .....	8
3. Features .....	10
4. Terms & Concepts .....	11
4.1. Complex event processing.....	11
4.2. Microsoft StreamInsight .....	11
4.3. Adapter.....	12
4.4. Input Adapter.....	12
4.5. Output Adapter.....	12
4.6. Queries.....	12
5. Operating Systems Compatibility .....	12
6. OPC Compatibility .....	12
7. StreamInsight Compatibility .....	12
8. System Requirements.....	13
<b>INSTALLING OPC ADAPTER FOR MICROSOFT STREAMINSIGHT .....</b>	<b>14</b>
1. Pre-Installation Considerations .....	14
2. Installing and Running .....	14
3. Files Included in the Distribution .....	22
4. Removing the OPC Adapter for Microsoft StreamInsight .....	23
<b>INTERACTING WITH THE IO OPC STREAMINSIGHT INPUT ADAPTER .....</b>	<b>25</b>
1. IntegrationObjects.OPCDA.InputAdapterManager Namespace .....	25
1.1. OPCDAEvent Class .....	25
1.2. OPCDAEventInputConfig Class.....	26
1.3. OPCDAEventInputFactory Class.....	27
1.4. DeclareAdvanceTimeProperties Method .....	28
2. IntegrationObjects.OPCDA.Data Namespace .....	28
2.1. Document Class.....	28
2.2. OPCAddressSpace Class.....	29

2.3. OPCServer Class.....	29
3. IntegrationObjects.OPCDA.Tools Namespace.....	30
3.1. OPCServerManager Class .....	30
3.2. StreamInsightManager Class.....	34
<b>INTERACTING WITH THE IO STREAMINSIGHT OUTPUT ADAPTER.....</b>	<b>35</b>
1. IntegrationObjects.OPCDA.OutputAdapterManager Namespace.....	35
1.1. OutputAdapterConfiguration Class .....	35
1.2. OPCDAEventOutputFactory Class .....	35
2. IntegrationObjects.OPCDA.OutputAdapter.Data Namespace .....	36
2.1. DataEvent Class .....	36
2.2. DataEntity Class .....	36
<b>CREATING A STREAMINSIGHT APPLICATION .....</b>	<b>38</b>
1. Compiling and Linking.....	38
1.1. Step1: Create your Visual Studio Project .....	38
1.2. Step2: Select the Target .NET Framework.....	39
1.3. Step3: Add the Required References .....	40
2. Implementing a StreamInsight Application .....	42
2.1. Create the User Interface.....	42
2.2. Add Using Directives, Parameters and Main Class .....	43
2.3. Configure OPC Adapter for Microsoft StreamInsights .....	44
2.4. Configure OPC StreamInsight Input Adapter.....	45
2.5. Define the Query.....	46
2.6. Configure OPC StreamInsight Output Adapter .....	46
2.7. Creating the CEPQuery .....	48
2.8. Starting the Query .....	48
2.9. Stopping the Query .....	50
<b>USING IO STREAMINSIGHT SAMPLES.....</b>	<b>51</b>
1. Overview.....	51
2. OPC StreamInsight WinForms Sample .....	51
2.1. Open the Sample with Visual Studio2010 .....	51
2.2. Add the Required References.....	52
2.3. Generate CSV File .....	53
2.4. Configure the Input Adapter .....	56
2.5. Configure the Output Adapter .....	57
2.6. Configure Instance and Application Names.....	57
2.7. Start the StreamInsight Application .....	58
2.8. Stop the StreamInsight Application.....	59
3. OPC StreamInsight Console Sample .....	59
4. How to Use the OPC Adapters with StreamInsight 2.0.....	62

<b>TRACING CAPABILITIES .....</b>	<b>64</b>
1. Overview.....	64
2. Sample Configuration File.....	66
<b>GLOSSARY .....</b>	<b>67</b>

## TABLE OF FIGURES

Figure 1: StreamInsight Architecture.....	9
Figure 2: Integration Objects' OPC Adapter for Microsoft StreamInsight Architecture .....	10
Figure 3: Installation Welcome Dialog.....	15
Figure 4: License Agreement Dialog.....	16
Figure 5 : Customer Information Dialog .....	17
Figure 6: Setup Type Dialog .....	18
Figure 7: Choose Destination Folder Dialog .....	19
Figure 8: Custom Installation Dialog .....	20
Figure 9: Installation Dialog .....	21
Figure 10: Installation Completed Dialog .....	22
Figure 11: Uninstaller Icon in the start menu.....	23
Figure 12: Uninstall the OPC Adapter for Microsoft StreamInsight.....	23
Figure 13: New Project Window.....	38
Figure 14: Project Explorer Window.....	39
Figure 15: Select .NET framework .....	39
Figure 16: Add Reference Menu .....	40
Figure 17: Adding Reference Dialog.....	41
Figure 18: OPC Adapter for Microsoft StreamInsight References .....	41
Figure 19: Graphical User Interface .....	42
Figure 20: Project Explorer .....	52
Figure 21: Add Reference.....	53
Figure 22: Main Dialog of the sample .....	54
Figure 23: Configuration File Generator Window .....	54
Figure 24: Select folder Dialog.....	55
Figure 25: Example of Configuration File .....	56
Figure 26: Input Adapter Configuration Window.....	56
Figure 27: Output Adapter Configuration Window.....	57
Figure 28: StreamInsight Application Configuration .....	58

Figure 29: Displayed OPC real-time data.....	59
Figure 30: OPC StreamInsight Console Sample .....	60
Figure 31: Configuration Dialog Box .....	61
Figure 32 : Displaying Data in Console .....	62

## LIST OF TABLES

Table 1: Files included in the Distribution.....	23
Table 2: Attributes of OPCDAEvent Class .....	26
Table 3: Parameters of the Init Method .....	26
Table 4: Attributes of the OPCDAEventInputConfig Class .....	27
Table 5: Parameters of the Create Method .....	28
Table 6: Parameters of the DeclareAdvanceTimeProperties Method.....	28
Table 7: Document Class Attributes.....	29
Table 8: Attributes of the OPCAddressSpace Class .....	29
Table 9: Attributes of the OPCServer Class.....	30
Table 10: Parameters of the ListServers Method .....	30
Table 11: Parameters of the Browse Method.....	30
Table 12: Parameters of the BrowseSourceFlat Method.....	31
Table 13: Parameters of the ConnectOPCServer Method .....	31
Table 14: Parameters of the QueryOrganization Method .....	32
Table 15: Parameters of the ChangeBrowsePosition Method.....	32
Table 16: Parameters of the BrowseOPCItemIDs Method .....	33
Table 17: Parameters of the GetItemID Method.....	33
Table 18: Parameters of the SaveCSVFile Method.....	33
Table 19: CSV Configuration File Format .....	34
Table 20: Parameters of the getStreamInsightInstances Method.....	34
Table 22: Parameters of the SubscribeDataChangeEvent Method .....	35
Table 23: Attributes of the OPCDAEventOutputFactory Class .....	36
Table 24: Parameters of the DataEvent Method .....	36
Table 25: Attributes of the DataEntity Class.....	37
Table 26: Log Settings.....	65

# PREFACE


## About This User Guide

This guide describes how to use the OPC Adapter for Microsoft StreamInsight and provides instructions and code samples that explain how to configure the adapters, build queries, and stream event points. It also includes hands-on tutorials to help you quickly learn these concepts.

## Target Audience

This document is intended for Integration Objects' OPC Adapter for Microsoft StreamInsight application developers. Basic knowledge of the .NET framework, C# .NET language, and StreamInsight is assumed.

## Document Conventions

Convention	Description
<b>Bold</b>	Click/selection action required.
<i>Blue bold italics</i>	Reference to other sections, or to other Integration Objects user guides.
	Information to be noted.
Consolas type	C# .NET method signatures.
Lucida Console Type	Reference to configuration ini files.

## Customer Support Services

Offices	Email
<b>Houston, USA:</b> +1 713 609 9208	Support: <a href="mailto:customerservice@integrationobjects.com">customerservice@integrationobjects.com</a>
<b>Genova, Italy:</b> +39 34 75 83 93 47	Sales: <a href="mailto:sales@integrationobjects.com">sales@integrationobjects.com</a>
<b>Tunis, Tunisia:</b> +216 71 195 360	Online: <a href="http://www.integrationobjects.com">www.integrationobjects.com</a>

# INTRODUCTION

## 1. Overview

Integration Objects' OPC Adapter for Microsoft StreamInsight includes a set of .NET APIs that integrate an OPC Client module with Microsoft StreamInsight. StreamInsight is a platform for developing Complex Event Processing (CEP) applications. With OPC Adapter for Microsoft StreamInsight, users can develop CEP applications, allowing them to write analytical queries against OPC real-time data that monitor, analyze, report, record, and filter OPC data events.

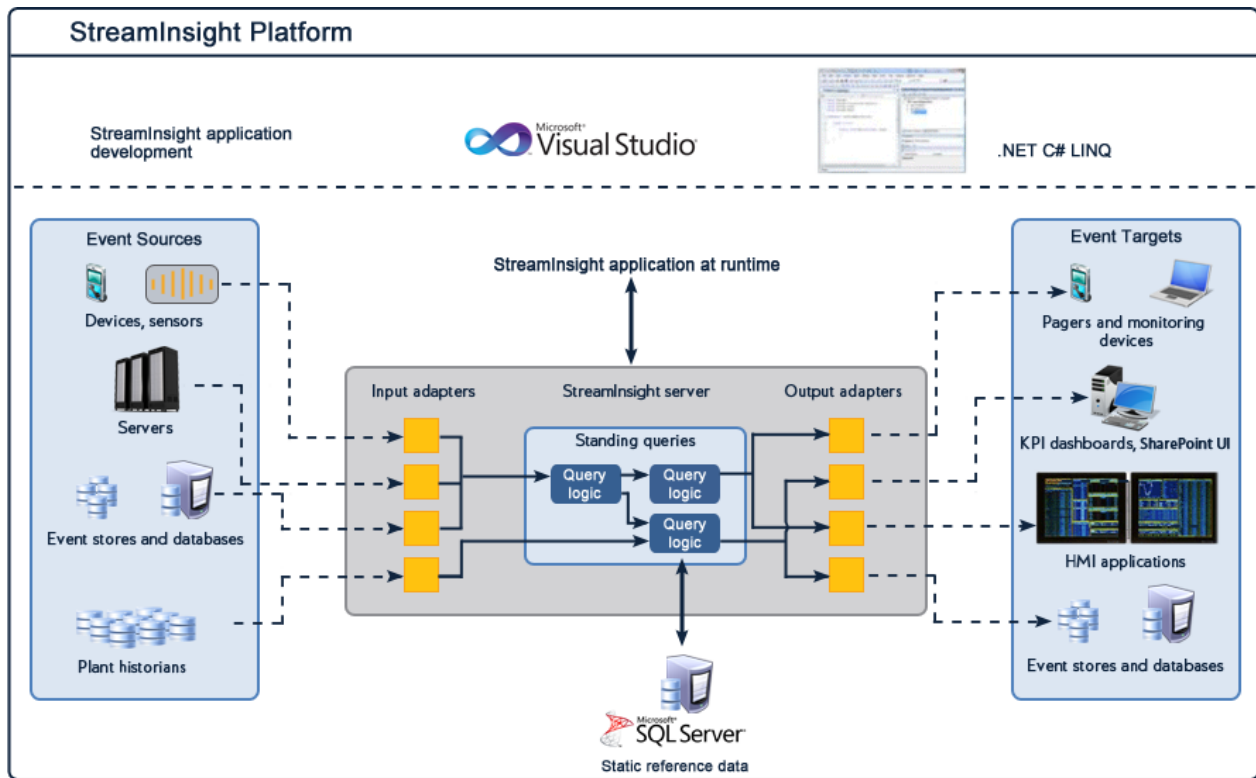
Real-time process data collected from OPC DA Servers can be streamed into the StreamInsight platform as events using the OPC StreamInsight input adapter. Then, the events generated by StreamInsight can be streamed out of it using any output adapter or the output adapter included within the OPC Adapter for Microsoft StreamInsight package.

While the OPC StreamInsight input adapter streams events from OPC Servers into StreamInsight, the output adapter handles events to the client application. These adapters are provided as assemblies that developers can reference to and use in their queries which can be defined using the Language Integrated Queries (LINQ) with Microsoft StreamInsight.

## 2. Architecture

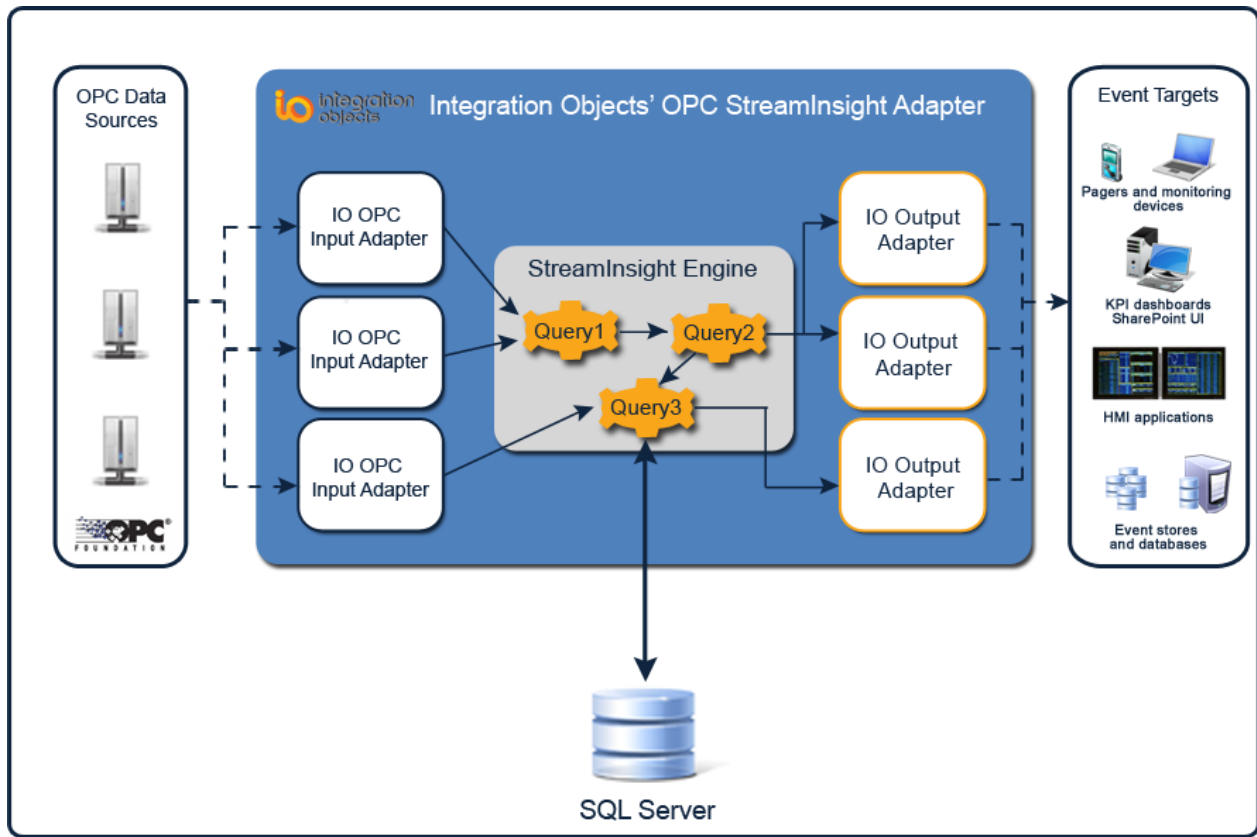
OPC Adapter for Microsoft StreamInsight interacts with the StreamInsight platform by exchanging events.





**Figure 1: StreamInsight Architecture**

Integration Objects' OPC Adapter for Microsoft StreamInsight product includes adapters that provide standardized connectivity to the StreamInsight low-latency complex event processing (CEP) platform.



**Figure 2: Integration Objects' OPC Adapter for Microsoft StreamInsight Architecture**

Using Integration Objects' OPC Adapter for Microsoft StreamInsight, .NET custom applications will be able to access data from any OPC DA Server without having to be concerned about the details of the OPC standard interfaces or COM/DCOM, stream real-time data into Microsoft StreamInsight, and build queries that monitor, analyze, report, record, and filter real-time process data.

### 3. Features

The OPC Adapter for Microsoft StreamInsight offers the following features:

- **Seamless Integration to StreamInsight Platform:** The product includes both an OPC StreamInsight input adapter and a typical .NET output adapter, allowing you to build your CEP applications and integrate them easily to the StreamInsight platform. The input streams events from OPC Servers into StreamInsight and the output adapter handles events to the client application.
- **Reading real-time data from OPC DA Servers:** The OPC input adapter allows application developers to collect real-time data from multiple OPC Servers and OPC Items. The management of the OPC groups is completely transparent, reducing the number of objects that the developers have to handle. The input adapter also allows:
  - Performing synchronous/asynchronous reads of OPC Items values, timestamps, and qualities from OPC DA Servers.

- Browsing OPC Servers address space.
- **Ease of Configuration:** The input adapter is easy to configure by using a CSV configuration file where you can list the OPC Servers, Items and their settings. The sample included in the setup package has an intuitive graphical user interface that allows you to generate this CSV configuration file.
- **Automatic Reconnection:** The OPC input adapter manages multiple local and remote connections with OPC DA Servers. When a communication breaks or a connection is lost with an OPC DA Server, the OPC input adapter will automatically try to reconnect to this OPC Server and re-establish the communication.
- **Producing StreamInsight events:** Transforming OPC real-time data to StreamInsight events and managing their consumption.
- **Helpful log capabilities:** OPC Adapter for Microsoft StreamInsight performs extensive errors and messages tracking, thus offering the user many ways to monitor and troubleshoot running operations:
  - All log messages are saved in text files
  - The user is allowed to use five different log levels

## 4. Terms & Concepts

### 4.1. Complex event processing

Complex event processing (CEP) consists of processing many events happening across all the layers of an organization, identifying the most meaningful events within the event cloud, analyzing their impact, and taking subsequent action in real-time.

Complex event processing refers to process states, the changes of state exceeding a defined threshold level, time, or value increment or just a counter as events. It requires the respective event monitoring, event reporting, event recording, and event filtering. An event may be observed as a change of state or of a data value with any physical or logical or otherwise discriminated condition of in a system, each information state with an attached time stamp defining the order of occurrence and a topology mark defining the location of occurrence.

### 4.2. Microsoft StreamInsight

Microsoft StreamInsight provides a powerful platform for developing and deploying complex event processing (CEP) applications. CEP is a technology for high-throughput, low-latency processing of event streams. Typical event stream sources include data from manufacturing applications, control systems, financial trading applications, or operational analytics. The StreamInsight stream processing architecture and the familiar .NET-based development platform enable developers to quickly implement robust and highly efficient event processing applications.

### 4.3. Adapter

Adapter is a software transformer that delivers events into or out of the StreamInsight server.

### 4.4. Input Adapter

Input adapter is a software transformer that accepts incoming event/data streams from external sources such as OPC Servers. The input adapter reads the incoming events in the format in which they are supplied and transforms them into a format that is consumable by the StreamInsight server.

### 4.5. Output Adapter

Output adapter is a software transformer that receives events processed by the StreamInsight server, transforms the events into a format expected by the output device, and emits the data to that device. The output device may be a database, text file, PDA, GUI, or other devices.

### 4.6. Queries

Queries in StreamInsight are defined using Language Integrated Queries (LINQ) and operate as standing queries on event streams. A standing query operates on the values arriving in one or more streams by filtering them, adjusting their shape, performing joins over events that are valid at the same time, and projecting new payloads from the query into a result stream.

## 5. Operating Systems Compatibility

OPC Adapter for Microsoft StreamInsight supports the following operating systems:

- Windows XP Service Pack 2 or later (x86 and x64)
- Windows Server 2003 Service Pack 2 or later (x86 and x64)
- Windows Server 2008 (x86 and x64)
- Windows Server 2012 (x86 and x64)
- Windows 7 (x86 and x64)
- Windows 8 (x86 and x64)

## 6. OPC Compatibility

The OPC input adapter supports OPC Data Access version 1.0a, 2.05 and higher.

## 7. StreamInsight Compatibility

The OPC Adapter for Microsoft StreamInsight supports:

- StreamInsight version 1.2.
- StreamInsight version 2.0.
- StreamInsight version 2.1.
- StreamInsight version 2.3.

The support of higher versions will be explained later in this user guide.

## 8. System Requirements

The following are the system requirements to run the OPC Adapter for Microsoft StreamInsights:

- Recommended: 2.2 GHz or faster CPU, 1024 MB or more of RAM
- Minimum: 1.6 GHz CPU, 384 MB of RAM

# INSTALLING OPC ADAPTER FOR MICROSOFT STREAMINSIGHT

## 1. Pre-Installation Considerations

Before you run the OPC Adapter for Microsoft StreamInsight, verify that the target computers;

1. Meet the minimum system requirements. *Refer to section 8 of the previous chapter.*
2. Include an enabled .NET Framework 3.5 SP1. To be able to use the sample projects, you will need to install .NET Framework 4.

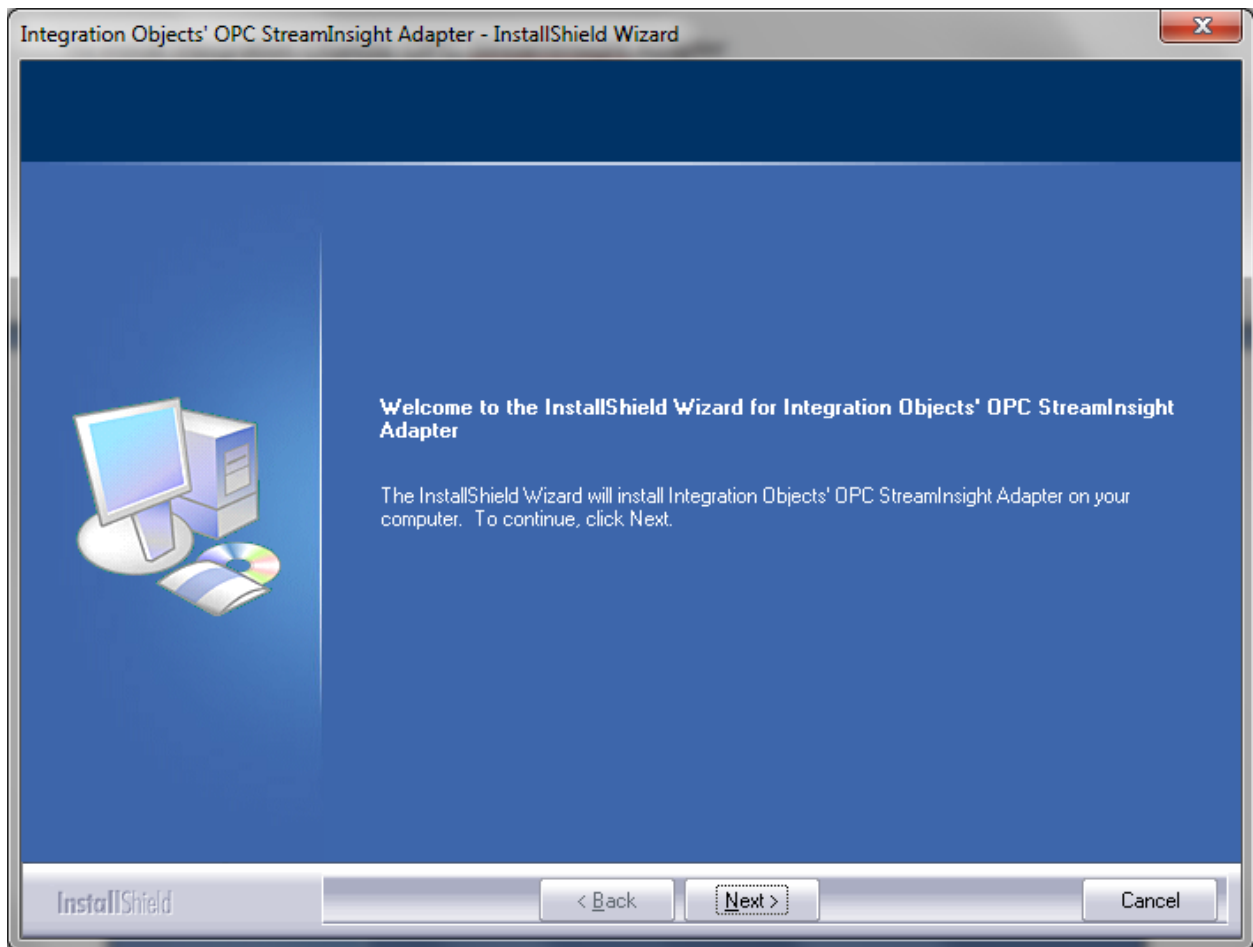
If you are using StreamInsight 2.0 or StreamInsight 2.1 or StreamInsight 2.3, you have to install .NET Framework 4.0.

3. Install StreamInsight 1.2 or higher.
4. Have the security permissions required to install software.
5. Include Microsoft Visual Studio if the system is to be used to develop StreamInsight applications.

## 2. Installing and Running

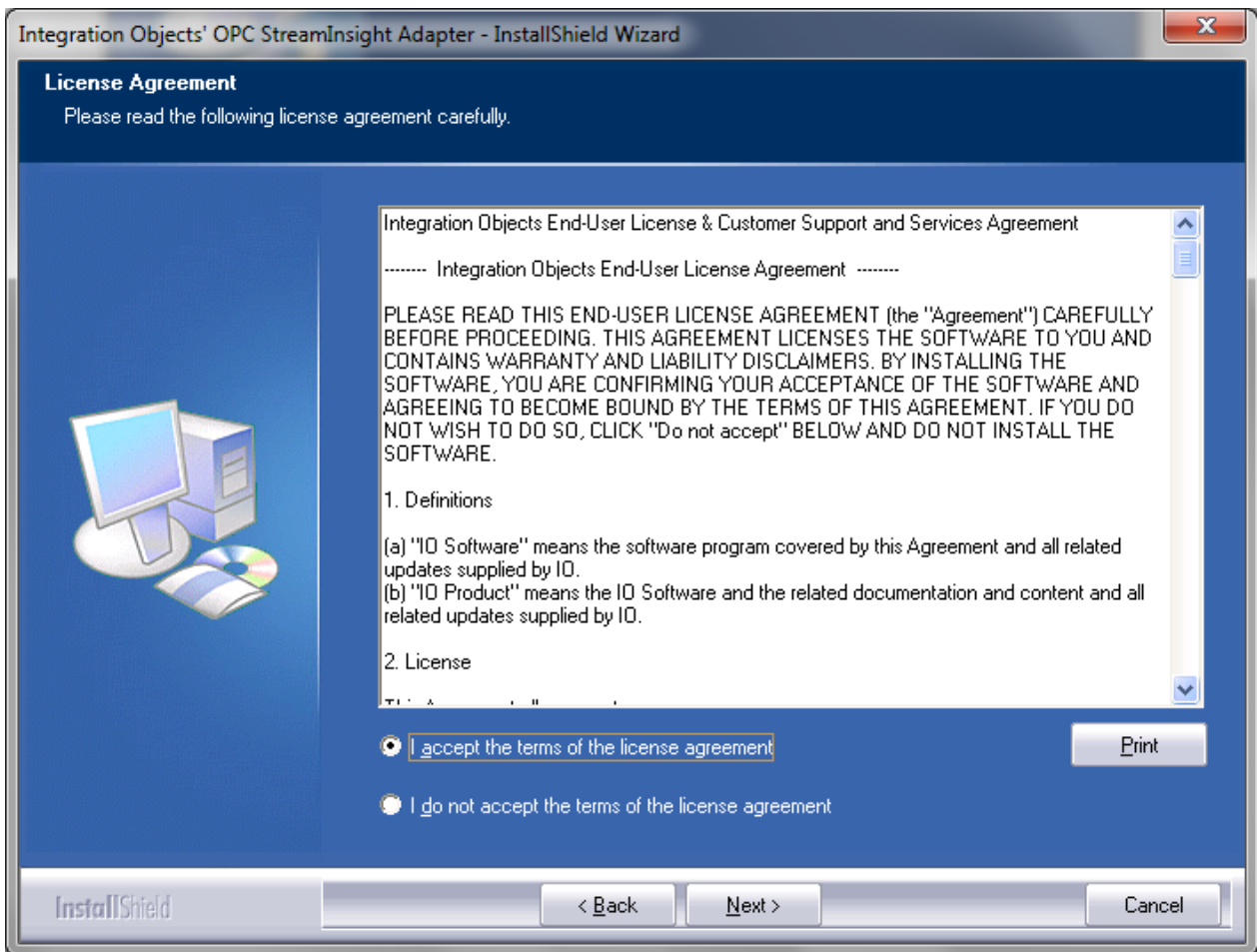
To install Integration Objects' OPC Adapter for Microsoft StreamInsight:

1. Double-click on the Integration Objects' OPC Adapter for Microsoft StreamInsight installation package. The installation welcome dialog box will appear.



**Figure 3: Installation Welcome Dialog**

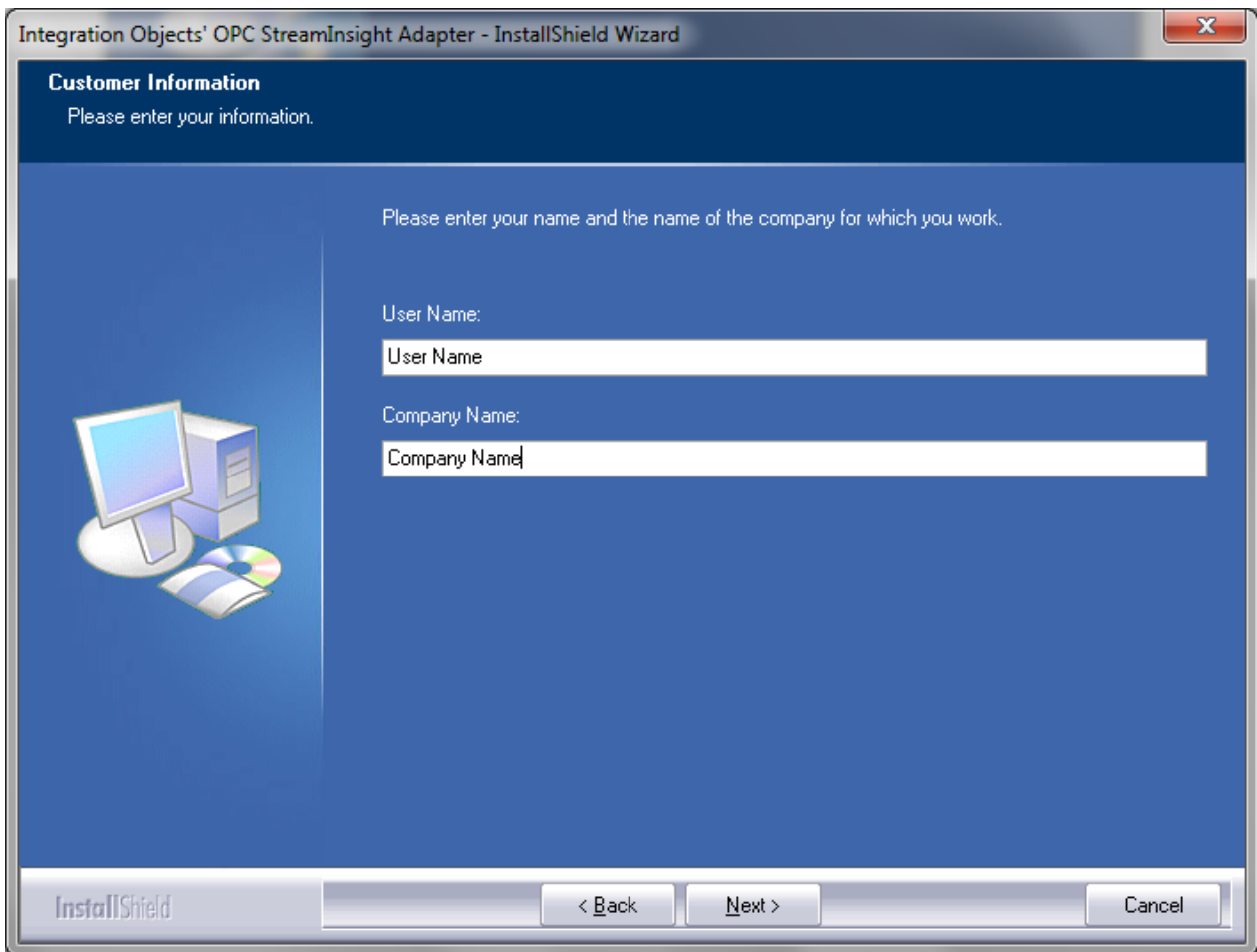
2. Click the **Next** button. The license agreement (Figure 4) will be displayed.



**Figure 4: License Agreement Dialog**

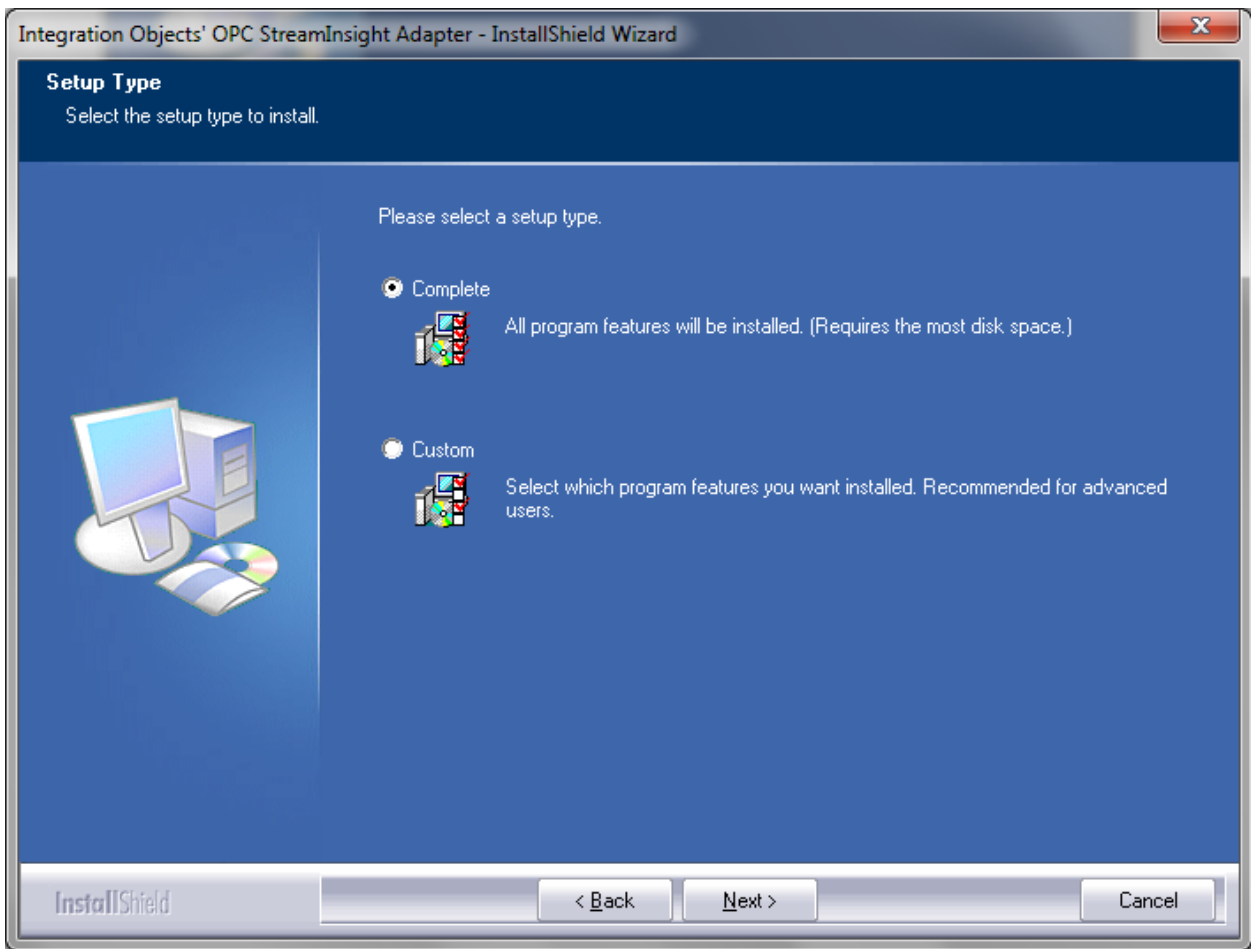
3. After reading the license agreement, select the first option and click the **Next** button. By proceeding, you are accepting all of the license agreement terms. Otherwise, you can cancel the installation. The customer information dialog box (Figure 5) will appear.





**Figure 5 : Customer Information Dialog**

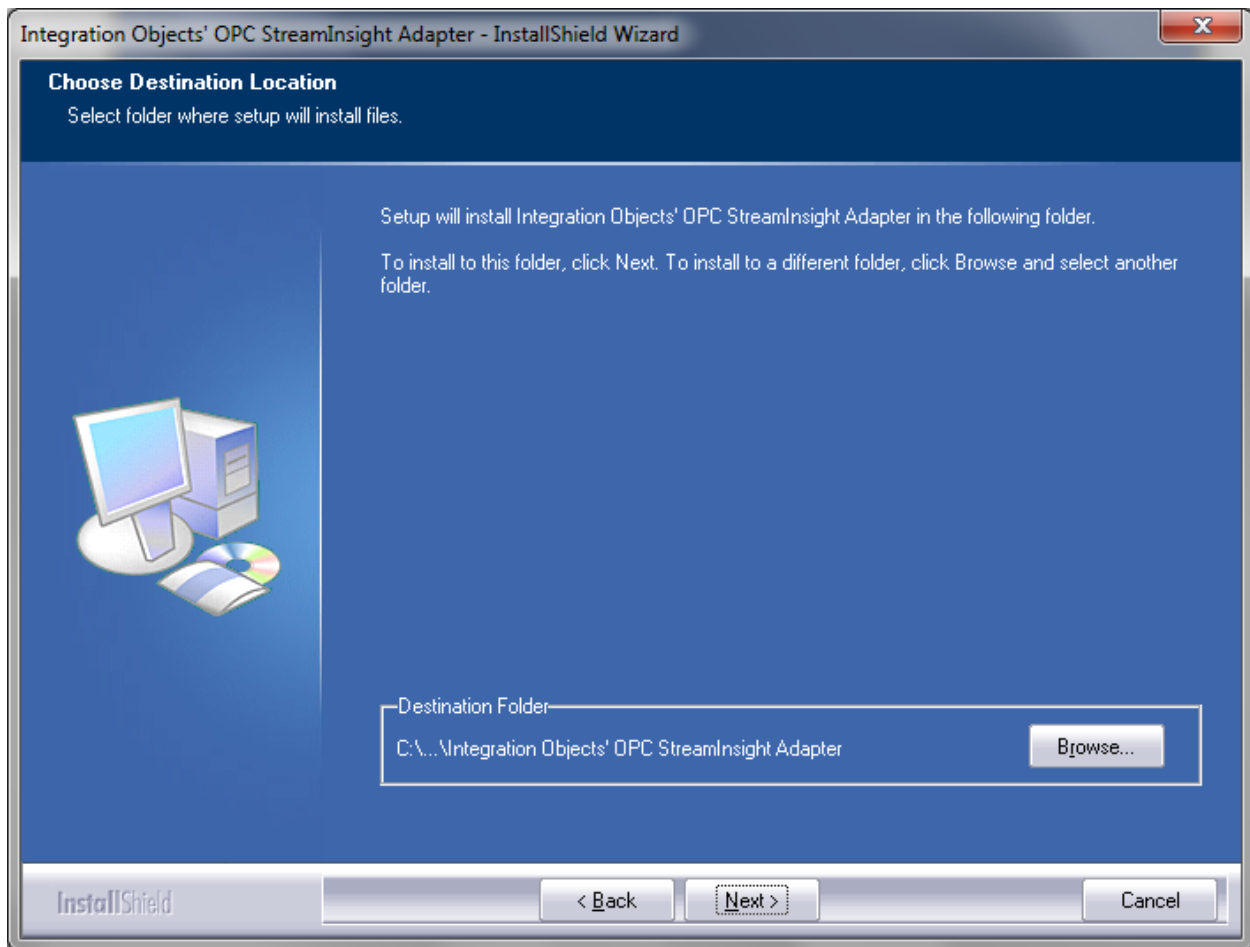
4. Enter your user name and company name and then click the **Next** button. The dialog box for choosing the type of your installation (Figure 6) will be displayed.



**Figure 6: Setup Type Dialog**

5. Select the installation type :
  - a. **Complete Type:** That will install all the components included in the setup.
  - b. **Custom Type:** In this installation type, the user will be able to select only features that he needs.

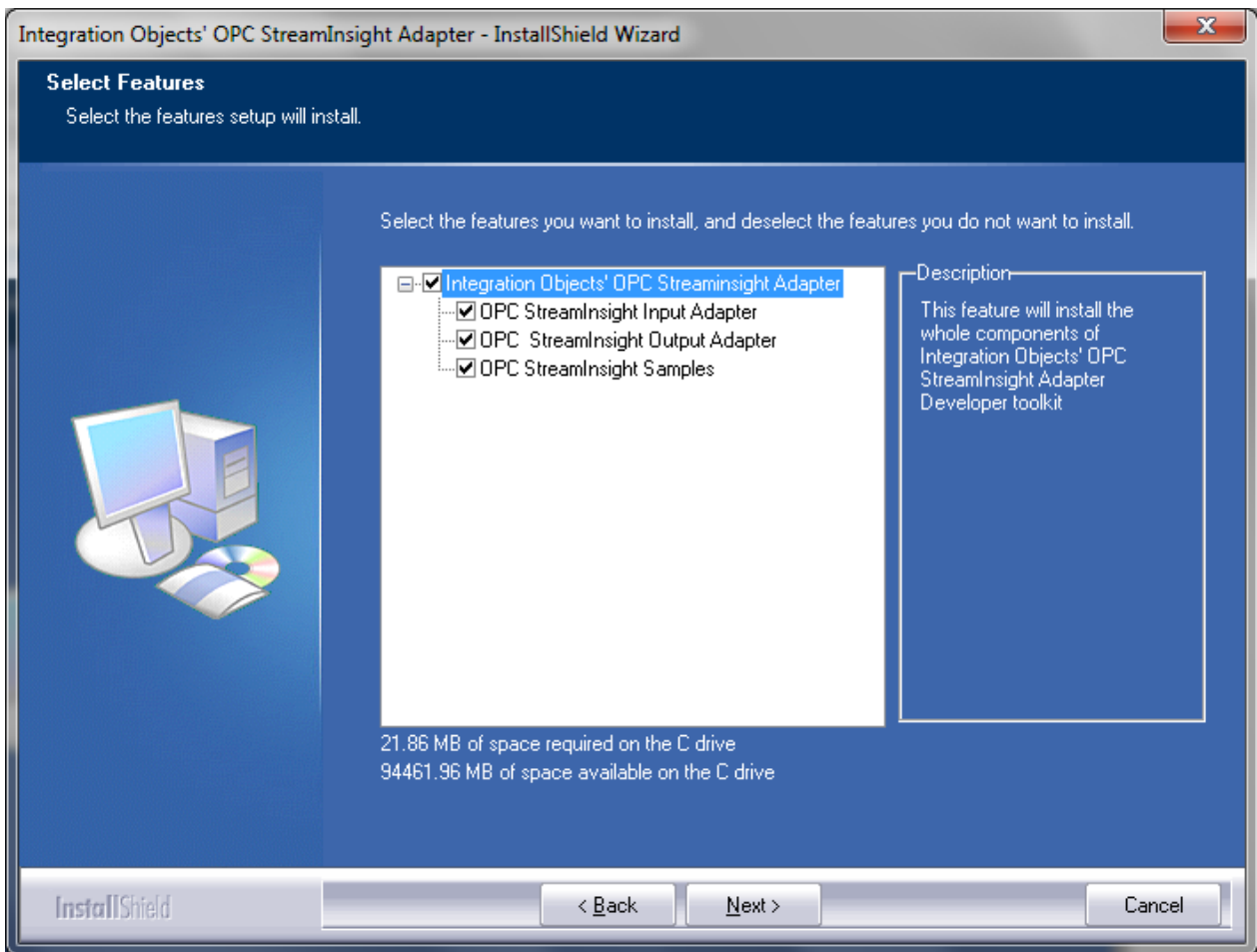
Select your option and click on **Next** button, the dialog box for choosing the destination folder will be displayed.



**Figure 7: Choose Destination Folder Dialog**

Click the **Next** button to continue the installation, or the **Browse** button to select a different destination folder.

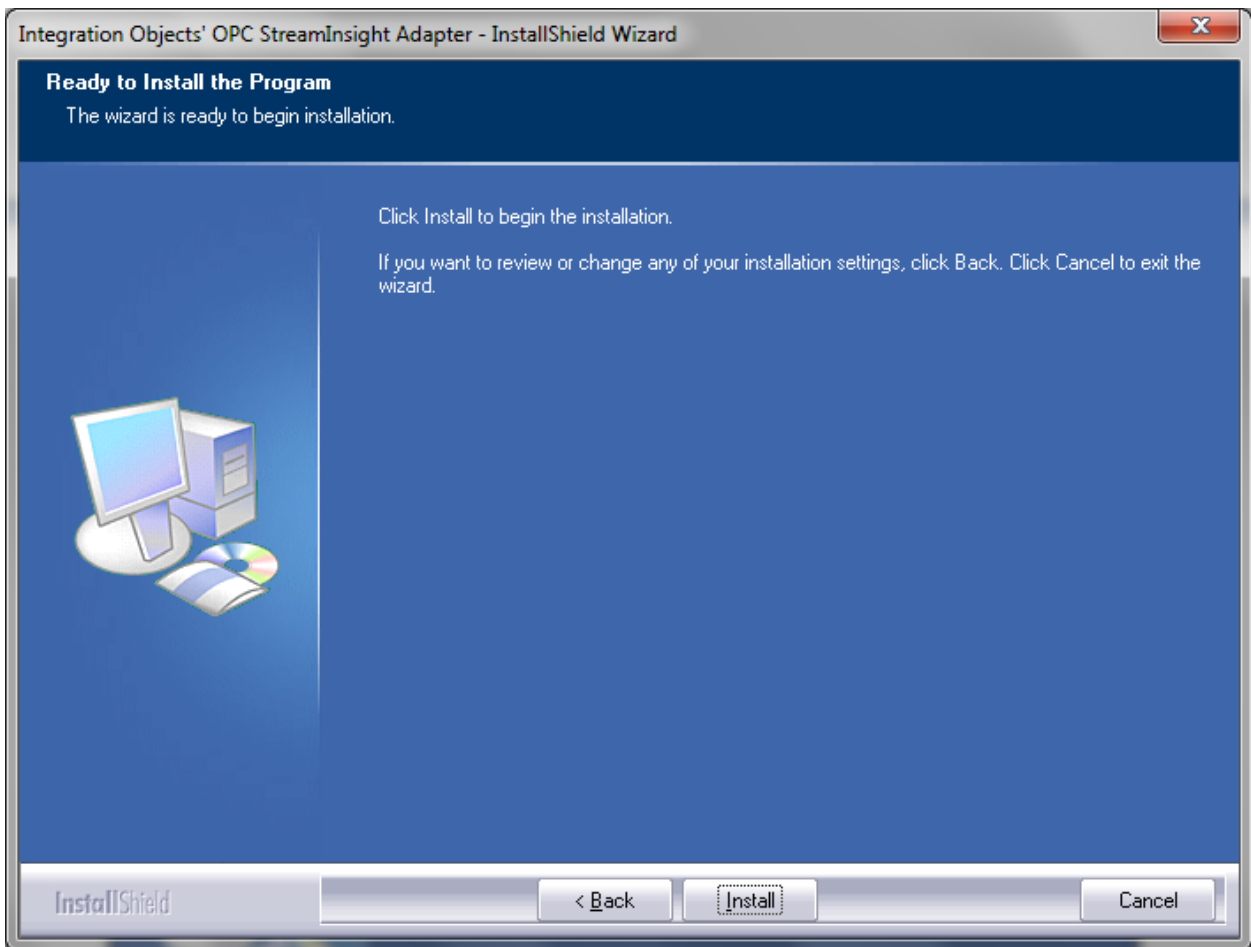
In this step, if the user previously selected the “Custom Installation Type”, the “Select Features” dialog (Figure 8) will appear. If this is not the case, the “Ready to Install the Program” dialog (Figure 9) will be displayed:



**Figure 8: Custom Installation Dialog**

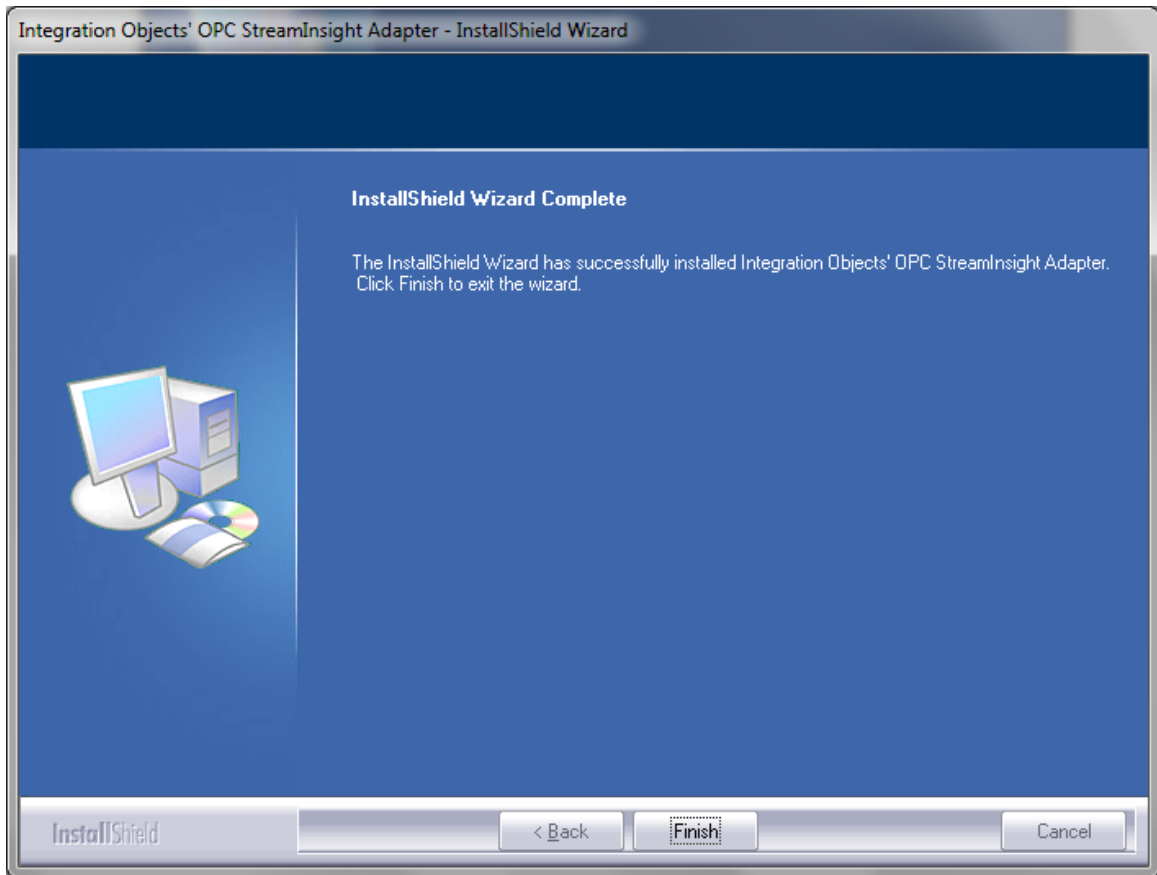
Figure 8 illustrates the different features that the user can install:

- OPC StreamInsight Input Adapter: This feature will install the OPC StreamInsight input adapter.
- OPC StreamInsight Output Adapter: This feature will install the OPC StreamInsight output adapter.
- OPC StreamInsight Samples: This feature will install Two OPC StreamInsight sample Visual Studio 2010 projects.



**Figure 9: Installation Dialog**

6. Click the Install button to start the installation. The setup will then:
  - Copy the necessary files to the selected target folder,
  - Create shortcut icons to launch the OPC Adapter for Microsoft StreamInsight sample from the desktop and an authorization license program from the start menu,
  - Make an un-installation entry in the Add/Remove Programs in the Control Panel.
7. Once the installation is completed, the following window will be displayed. Click the **Finish** button to complete the installation and close the wizard.



**Figure 10: Installation Completed Dialog**

### 3. Files Included in the Distribution

After the installation is completed, the following files and folders are deployed on your system and located under the installation folder of the OPC Adapter for Microsoft StreamInsight:

Folder	Installed files
<b>Bin</b>	It includes the following files and folders: <ul style="list-style-type: none"> <li>• <b>IntegrationObjects.Logger.SDK.dll</b>: The log toolkit used for tracing events and operations.</li> <li>• <b>OPC StreamInsight Input Adapter folder</b>: It contains the following files:               <ul style="list-style-type: none"> <li>▪ IntegrationObjects.OPCDA.InputAdapter.SDK.dll: The OPC StreamInsight input adapter .NET API.</li> <li>▪ OPCNetClientSDK.dll: Integration Objects' OPC core toolkit.</li> <li>▪ OpcRcw.Comn.dll: The OPC foundation SDK.</li> </ul> </li> <li>• <b>OPC StreamInsight Output Adapter folder</b>: it contains:               <ul style="list-style-type: none"> <li>▪ IntegrationObjects.OPCDA.OutputAdapter.SDK.dll: The OPC StreamInsight output adapter .NET API.</li> </ul> </li> </ul>

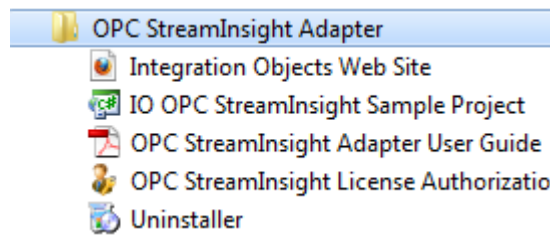
<b>Configuration Files</b>	It contains the .ini configuration files containing information about log settings.
<b>Doc</b>	This folder contains the OPC Adapter for Microsoft StreamInsight user guide.
<b>License Authorization</b>	This folder contains the license authorization application that will be used to register and activate the OPC Adapter for Microsoft StreamInsight product.
<b>Samples</b>	This folder contains two sample Visual Studio 2010 projects: a Windows Forms sample project and a console application sample project showing how to use the OPC Adapter for Microsoft StreamInsight.

**Table 1: Files included in the Distribution**

## 4. Removing the OPC Adapter for Microsoft StreamInsight

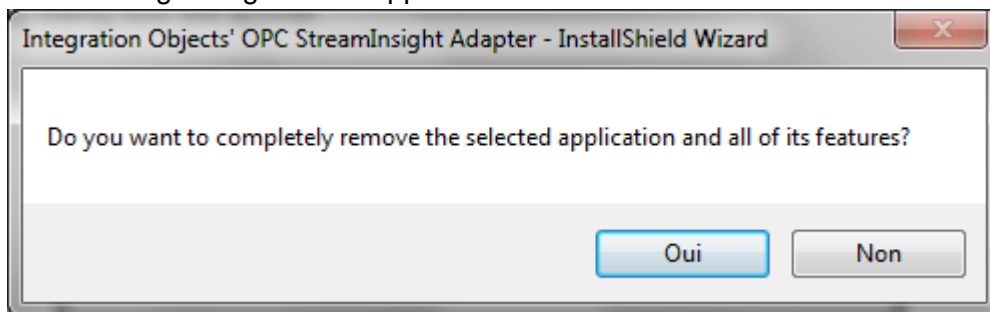
To uninstall the OPC Adapter for Microsoft StreamInsight, follow the steps below:

1. Click the **Uninstaller** shortcut icon in the start menu, as shown in Figure 11.



**Figure 11: Uninstaller Icon in the start menu**

The following dialog box will appear:



**Figure 12: Uninstall the OPC Adapter for Microsoft StreamInsight**

2. Click the **Yes** button to start uninstalling.
3. Click **Finish** when the un-installation is complete.

The OPC Adapter for Microsoft StreamInsight can also be removed manually as follows:

1. Go to the **Control Panel**.
2. Click **Add/Remove Programs**.
3. In the Add/Remove Programs dialog screen, select **Integration Objects' OPC Adapter for Microsoft StreamInsight**.

Click **Change/Remove** then **OK**.



# INTERACTING WITH THE IO OPC STREAMINSIGHT INPUT ADAPTER

In this chapter, we will describe all classes and methods of the OPC StreamInsight input adapter toolkit.

## 1. IntegrationObjects.OPCDA.InputAdapterManager Namespace

### 1.1. OPCDAEvent Class

The OPCDAEvent class is the OPC data stream event model. It is a typed class point input adapter. This class is to be sent in the payload of **PointEvent** into StreamInsight Engine. The **PointEvent** is the basic unit of data processed by the CEP server. Each event consists of the following parts:

- **Header:** An event header contains metadata that defines the event type and one or more timestamps that define the time interval of the event. The timestamps are supplied by the OPC Server source.
- **Payload:** The payload of an event is a .NET data structure that contains the OPC data associated with the event.

In this class you can find the following attributes:

Attribute	Type	Description
<b>hostname</b>	String	Name or IP address of the machine that hosts OPC DA Servers.
<b>ServerName</b>	String	ProgID of the OPC Server
<b>TagName</b>	String	The name of the OPC Item to be read and its data values to be sent to StreamInsight Engine.
<b>Quality</b>	Int	The OPC Item quality.
<b>Value</b>	Object	The data value of the OPC Item. This value is set as "Object" to support any OPC Item data type.

<b>TimeStamp</b>	String	The timestamp when the OPC data was received from the Server.
------------------	--------	---

**Table 2: Attributes of OPCDAEvent Class**

This class contains only one method which is the “Init” method used to initialize the OPCDAEvent class properties.

The following table describes the parameters of the “Init” method.

```
void Init(string hostName, string serverName, string tagName, short Quality, object value, string timeStamp);
```

In/Out	Parameter	Description
In	<b>hostName</b>	Name of the machine that hosts OPC DA Servers.
In	<b>ServerName</b>	ProgID of the OPC Server
In	<b>TagName</b>	The name of the OPC Item to be read and its data values to be sent to StreamInsight Engine.
In	<b>Quality</b>	The OPC Item quality.
In	<b>Value</b>	The data value of the OPC Item. This value is set as object to support any OPC Item data type.
In	<b>timeStamp</b>	The TimeStamp when the OPC data was received from the OPC Server.

**Table 3: Parameters of the Init Method**

## 1.2. OPCDAEventInputConfig Class

This class configures StreamInsight event production. It contains specific information about your input adapter instance. It is a mandatory class in the OPC StreamInsight input adapter.

It has the attributes described in the table below:

Attribute	Type	Description
<b>CtiFrequency</b>	uint	The current time increment which is used to sort the traffic.
<b>Ticks</b>	Double	The duration (in milliseconds) to start the event production.

<b>ConfigurationPath</b>	String	The path of the OPC StreamInsight input adapter CSV configuration file.
<b>MaxSendTries</b>	Int	The number of attempts the adapter will try to send an event before the operation will be declared as failed.
<b>ReconnectionTime</b>	Int	The time period that separates two successive reconnection attempts to the OPC Servers after a break in the communication was detected.
<b>DataProductionRate</b>	Int	Data events production rate in milliseconds.

**Table 4: Attributes of the OPCDAEventInputConfig Class**

### 1.3. OPCDAEventInputFactory Class

The OPCDAEventInputFactory class is the class responsible for creating instances of the adapter class. Depending on the events model (point, interval, or edge events), the factory class will instantiate the correct adapter class.

This class has the following methods:

#### 1.3.1. Create Method

This method creates the input adapter object. The following table describes the parameters of the “Create” method.

`InputAdapterBase Create<TPayload>(OPCDAEventInputConfig configInfo, EventShape eventShape)`

In/Out	Parameter	Description
In	<b>ConfigInfo</b>	Contains the input adapter configuration. It is an instance of the OPCDAEventInputConfig Class.
In	<b>EventShape</b>	Specifies the shape of events that are expected by the adapter. It could be: <ul style="list-style-type: none"> <li>• <b>Interval:</b> Interval events have a start time and an end time, indicating the lifetime during which the payload of the event is valid.</li> <li>• <b>Edge:</b> Edge events indicate either the start or the end of the lifetime of a payload. Events of Edge shape can be either start edges or end edges.</li> <li>• <b>Point:</b> Point events have a lifetime of a single tick and hence only a single</li> </ul>

		timestamp.
--	--	------------

**Table 5: Parameters of the Create Method**

The method returns a typed input adapter that will be responsible of streaming the data to the StreamInsight Engine.

## 1.4. DeclareAdvanceTimeProperties Method

This method declares the **AdvanceTime** properties by passing as input the configuration information and the shape of the event. This method belongs to Microsoft StreamInsight Adapter's interface **ITypedDeclareAdvanceTimeProperties**. This interface must be implemented by typed input factory classes that optionally want to specify **AdvanceTimeSettings** for the adapter instances that they produce.

Factory classes that implement this interface inform the runtime that it should produce current time increment (Cti) events that have the properties described by the **AdvanceTimeSettings** class returned by this method. The following table describes the different parameters of the "DeclareAdvanceTimeProperties" method:

```
AdapterAdvanceTimeSettings DeclareAdvanceTimeProperties <TPayload>
(OPCDAEventInputConfig configInfo, EventShape eventShape)
```

In/Out	Parameter	Description
In	<b>configInfo</b>	Configuration information for the adapter. It is an instance of the OPCDAEventInputConfig Class.
In	<b>eventShape</b>	Shape of the events that must be produced by the adapter.

**Table 6: Parameters of the DeclareAdvanceTimeProperties Method**

This method returns an instance of AdvanceTimeSettings object.

## 2. IntegrationObjects.OPCDA.Data Namespace

This namespace contains the data entity model of the OPC StreamInsight input adapter.

### 2.1. Document Class

The document class contains the main information of the CSV configuration file. The document class overrides the Equals method to give the possibility to the user to compare two document objects.

In the following table, you will find all the attributes of the class and their description:

Attribute	Type	Description
<b>Host_Name</b>	String	The name or the IP address of the OPC server machine.
<b>Server_Name</b>	String	The OPC Server ProgID.
<b>Item_Name</b>	String	The OPC Item name.
<b>Type</b>	String	The data type of the OPC Item.
<b>UpdateRate</b>	Int	The update rate at which data changes will be sent to the OPC input adapter. This value should be specified in milliseconds.
<b>Synchronous</b>	bool	This flag indicates if the tag will be read in synchronous mode or not.
<b>Alias</b>	String	The user can give an alias to the OPC Item to set a friendlier name and make it easier to recognize.

**Table 7: Document Class Attributes**

## 2.2. OPCAddressSpace Class

This class represents an OPC Server address space root node.

It contains the following attributes:

Attribute	Type	Description
<b>Name</b>	String	The name of the node which represents a branch in the OPC Server address space.
<b>Nodes</b>	ArrayList	A node can have children nodes. If it's the case, the children nodes are saved in an ArrayList. Otherwise, this property will be set to null.

**Table 8: Attributes of the OPCAddressSpace Class**

## 2.3. OPCServer Class

This class represents an OPC Server. You can find a description of its attributes in the following table:

Attribute	Type	Description
<b>Name</b>	String	The OPC Server ProgID.
<b>IP</b>	String	The OPC Server machine name or IP Address.

<b>Separator</b>	String	The separator used for the OPC Server address space branches.
<b>Nodes</b>	ArrayList	The address space nodes of the OPC Server.

**Table 9: Attributes of the OPCServer Class**

### 3. IntegrationObjects.OPCDA.Tools Namespace

This namespace contains a couple of classes attended to help the developer browse OPC Servers, generate a CSV configuration file and save it.

#### 3.1. OPCServerManager Class

This class contains static methods implemented to list OPC Servers, browse their address space, select OPC Items, and save their configuration into a CSV file format.

##### 3.1.1. ListServers Method

This method returns the list of the OPC Servers available in the specified machine in an ArrayList of OPCServer objects. The following table describes the input parameters of the ListServers method:

```
public static ArrayList ListServers(string hostName);
```

In/Out	Parameter	Description
In	<b>HostName</b>	The name or the IP address of the machine.

**Table 10: Parameters of the ListServers Method**

##### 3.1.2. Browse Method

This method browses the address space of all OPC Servers in a specific machine and returns an ArrayList of OPC Server objects. The following table describes the parameters of the browse method:

```
public static ArrayList Browse(string HostName);
```

In/Out	Parameter	Description
In	<b>HostName</b>	The Name or the IP address of the server machine.

**Table 11: Parameters of the Browse Method**

##### 3.1.3. BrowseSourceFlat Method

This method browses a specific OPC Server in a specific machine and returns an OPC Server object containing the address space of that server or null in case of failure. The following table describes the parameters of the BrowseSourceFlat method:

```
public static OPCServer BrowseSourceFlat(string HostName, string
svrName);
```

In/Out	Parameter	Description
In	<b>HostName</b>	The name or the IP address of the server machine.
In	<b>Server Name</b>	The OPC Server ProgID.

**Table 12: Parameters of the BrowseSourceFlat Method**

### 3.1.4. ConnectOPCServer Method

This method establishes the connection to an OPC Server and returns an identifier of the connection with this OPC Server. The following table describes the parameters of this method:

```
public static int ConnectOPCServer(string HostName, string
ServerName);
```

In/Out	Parameter	Description
In	<b>HostName</b>	The name or the IP address of the server machine.
In	<b>Serve Name</b>	The OPC Server ProgID.

**Table 13: Parameters of the ConnectOPCServer Method**

### 3.1.5. QueryOrganization Method

This method returns the address space organization type of the specified OPC Server. It can be used by the user in implementing the browsing of the OPC Server address space. The following table describes the parameters of this method:

```
public static void QueryOrganization(int index, out int
pNameSpaceType);
```

In/Out	Parameter	Description
In	<b>serverindex</b>	The identifier of the connection with the OPC Server.
Out	<b>pNameSpaceType</b>	The address space organization type which can be OPC_NS_HIERARCHIAL or OPC_NS_FLAT.

**Table 14: Parameters of the QueryOrganization Method**

### 3.1.6. ChangeBrowsePosition Method

This method allows moving in the hierarchical address space of the OPC Server. It can be used by the user to implement the browsing of the OPC Server address space. The following table describes the parameters of this method:

```
public static void ChangeBrowsePosition(int index, int direction,
string branchName);
```

In/Out	Parameter	Description
In	<b>index</b>	The identifier of the connection with the OPC Server.
In	<b>direction</b>	OPC_BROWSE_UP or OPC_BROWSE_DOWN or OPC_BROWSE_TO
In	<b>branchName</b>	The name of the branch to move into.

**Table 15: Parameters of the ChangeBrowsePosition Method**

### 3.1.7. BrowseOPCItemIDs Method

This method returns a list of OPC Item IDs. It can be used by the user in implementing the browsing of the OPC Server address space. The following table describes the parameters of this method:

```
public static void BrowseOPCItemIDs(int index, int direction, string
filterCriteria, short filterDatatype, int accessRightsFilter, out
ArrayList stringEnum);
```

In/Out	Parameter	Description
In	<b>index</b>	The identifier of the connection with the OPC Server.
In	<b>direction</b>	OPC_BRANCH or OPC_LEAF or OPC_FLAT.
In	<b>filterCriteria</b>	A server specific filter string.
In	<b>filterDatatype</b>	Filters the returned list based in the available data types. VT_EMPTY indicates no filtering.
In	<b>accessRightsFilter</b>	Filter based on the access rights.
Out	<b>stringEnum</b>	The returned branches names.



**Table 16: Parameters of the BrowseOPCItemIDs Method**

### 3.1.8. GetItemID Method

This method provides a way to get fully qualified item identification. It can be used by the user in implementing the browsing of the OPC Server address space. The following table describes the parameters of this method:

```
public static void GetItemID(int index, string ItemDataID, out string szItemID);
```

In/Out	Parameter	Description
In	<b>index</b>	The identifier of the connection with the OPC Server.
In	<b>ItemDataID</b>	The branch name.
Out	<b>szItemID</b>	The returned ItemID.

**Table 17: Parameters of the GetItemID Method**

### 3.1.9. SaveCSVFile Method

This method saves a list of Document instances into a CSV configuration file.

```
public static void SaveCSVFile(List<Document> documents, string fileName)
```

The following table describes the parameters of the SaveCSVFile method:

In/Out	Parameter	Description
In	<b>documents</b>	List of document entities.
In	<b>fileName</b>	The file path where the configuration information will be saved.

**Table 18: Parameters of the SaveCSVFile Method**


**Note:** Integration Objects' OPC StreamInsight input adapter accepts the input data in the following CSV File format:

Column	Name	Description
1	<b>ServerHostIP</b>	The OPC Server machine name or IP address.

2	<b>ServerProgID</b>	The OPC Server ProgID.
3	<b>TagName</b>	The OPC Item name.
4	<b>UpdateRate</b>	The update rate of the OPC Item in milliseconds.
5	<b>TagType</b>	The data type of the OPC Item.
6	<b>IsSynchronous</b>	A boolean to indicate whether the read mode is Synchronous or OnDataChange.
7	<b>AliasName</b>	The alias name allocated to the OPC Item.

**Table 19: CSV Configuration File Format**

### 3.2. StreamInsightManager Class

This class contains only one method which is the “getStreamInsightInstances” method.

This method returns a list that contains string values representing the StreamInsight instance names installed in the machine.

```
public static List<string> GetStreamInsightInstances(out string
strError)
```

The following table describes the parameters of the “getStreamInsightInstances” method:

In/Out	Parameter	Description
Out	<b>strError</b>	A string containing the error text if an error occurs.

**Table 20: Parameters of the getStreamInsightInstances Method**

# INTERACTING WITH THE IO STREAMINSIGHT OUTPUT ADAPTER

In this chapter, we will describe all classes and methods of the StreamInsight output adapter toolkit.

## 1. IntegrationObjects.OPCDA.OutputAdapterManager Namespace

### 1.1. OutputAdapterConfiguration Class

This class configures StreamInsight event consumption. It has one method **SubscribeToDataChangeEvent** which allows the end-user to subscribe to a DataEvent and asynchronously receive the OPC data stream events from the StreamInsight Engine.

```
public DataEvent SubscribeToDataChangeEvent ()
```

In/Out	Parameter	Description
Out	<b>DataEvent</b>	The received data events from the OPC Server.

**Table 21: Parameters of the SubscribeDataChangeEvent Method**

### 1.2. OPCDAEventOutputFactory Class

The factory class implements the IOutputAdapterFactory interface and its role consists in creating the output adapter objects. This class contains the “Create” method. The following table describes the parameters of this method:

```
public OutputAdapterBase Create(OutputAdapterConfiguration  
outputConfiguration, EventShape Shape, CepEventType EventType)
```

In/Out	Parameter	Description
In	<b>outputConfugration</b>	An instance of the outputAdapterConfiguration class.
In	<b>EventShape</b>	Specifies the shape of events that the stream contains. As for the input adapter, It could be: <ul style="list-style-type: none"> <li>• <b>Interval:</b> Interval events have a start time and an end time, indicating the lifetime during which the</li> </ul>

		payload of the event is valid. <ul style="list-style-type: none"> <li>• <b>Edge:</b> Edge events indicate either the start or the end of the lifetime of a payload. Events of edge shape can be either start edges or end edges.</li> <li>• <b>Point:</b> Point events have a lifetime of a single tick and hence only a single timestamp.</li> </ul>
In	<b>EventType</b>	Specifies the event type that is used by CEP application.

**Table 22: Attributes of the OPCDAEventOutputFactory Class**

This method returns an OutputAdapterBase class instance of the base class of OutputAdapters. The returned instance will be responsible of de-queuing the data from StreamInsight Engine.

## 2. IntegrationObjects.OPCDA.OutputAdapter.Data Namespace

### 2.1. DataEvent Class

The DataEvent class manages the events release. It contains the event handler to which the application should subscribe to receive the StreamInsight stream data. The following table contains a description of the events implemented in the DataEvent class:

```
private static event DataChangeEventHandler _dataChange;
private static event ShutdownAdapterHandler _shutdown ;
```

Event	Description
<b>DataChangeEventHandler</b>	The event that will be triggered upon the reception of new Streamed Data.
<b>ShutdownAdapterHandler</b>	The event that will be triggered upon a shutdown request from the user.

**Table 23: Parameters of the DataEvent Method**

### 2.2. DataEntity Class

The DataEntity class represents the StreamInsight event data container. When the output adapter consumes the StreamInsight events, it will transform them on DataEvent objects. The DataEvent will transport the DataEntity object received from StreamInsight Engine. This class has the following attributes:

Parameter	Type	Description
<b>HostName</b>	String	The name or the IP address of the OPC Server machine.
<b>SITimeStamp</b>	DateTimeOffset	The timestamp when the data was enqueued into StreamInsight Engine.

<b>serverName</b>	String	The OPC Server ProgID.
<b>tagName</b>	String	The name of the OPC Item received from the StreamInsight Engine.
<b>Quality</b>	String	The quality the OPC Item data value. It is represented as a string describing the received quality.
<b>Value</b>	Object	The OPC Item data value. This value is set as an object to support any OPC Item data type.
<b>TimeStamp</b>	String	The Timestamp when the OPC value was received from the OPC server.

**Table 24: Attributes of the DataEntity Class**

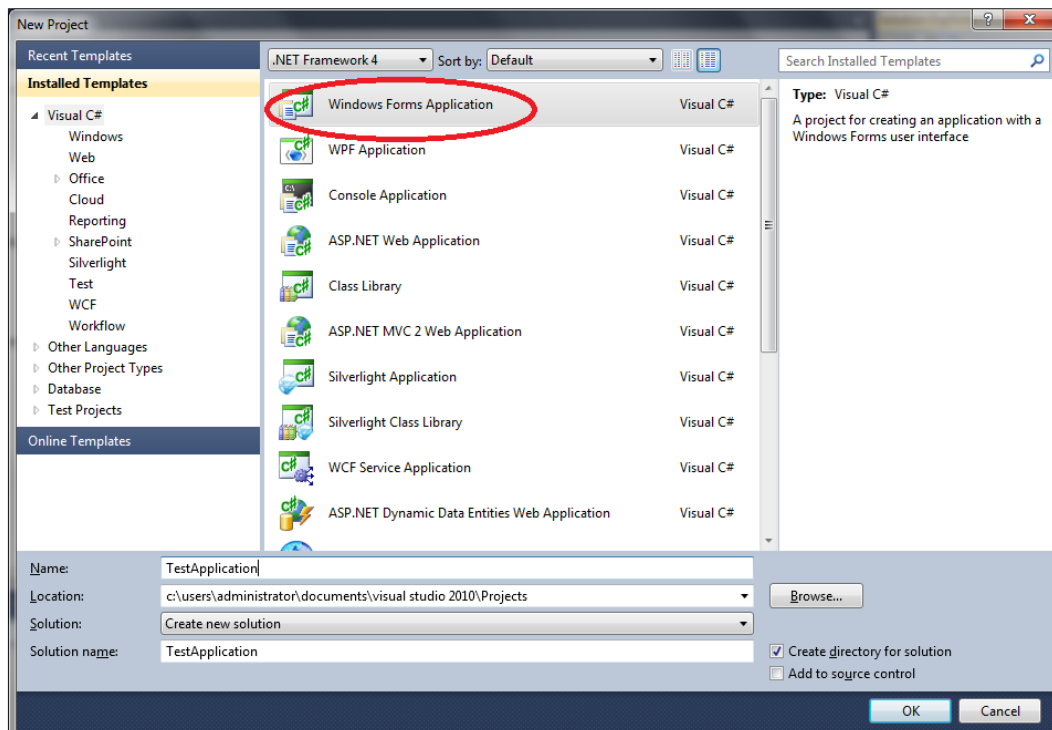
# CREATING A STREAMINSIGHT APPLICATION

## 1. Compiling and Linking

This section focuses on the steps to compile and correctly link your projects to develop a custom CEP application using Integration Objects' OPC Adapter for Microsoft StreamInsight, Microsoft StreamInsight Server, and Microsoft Visual Studio 2010.

### 1.1. Step1: Create your Visual Studio Project

Start Visual Studio2010 and choose New Project. The following window will be displayed.

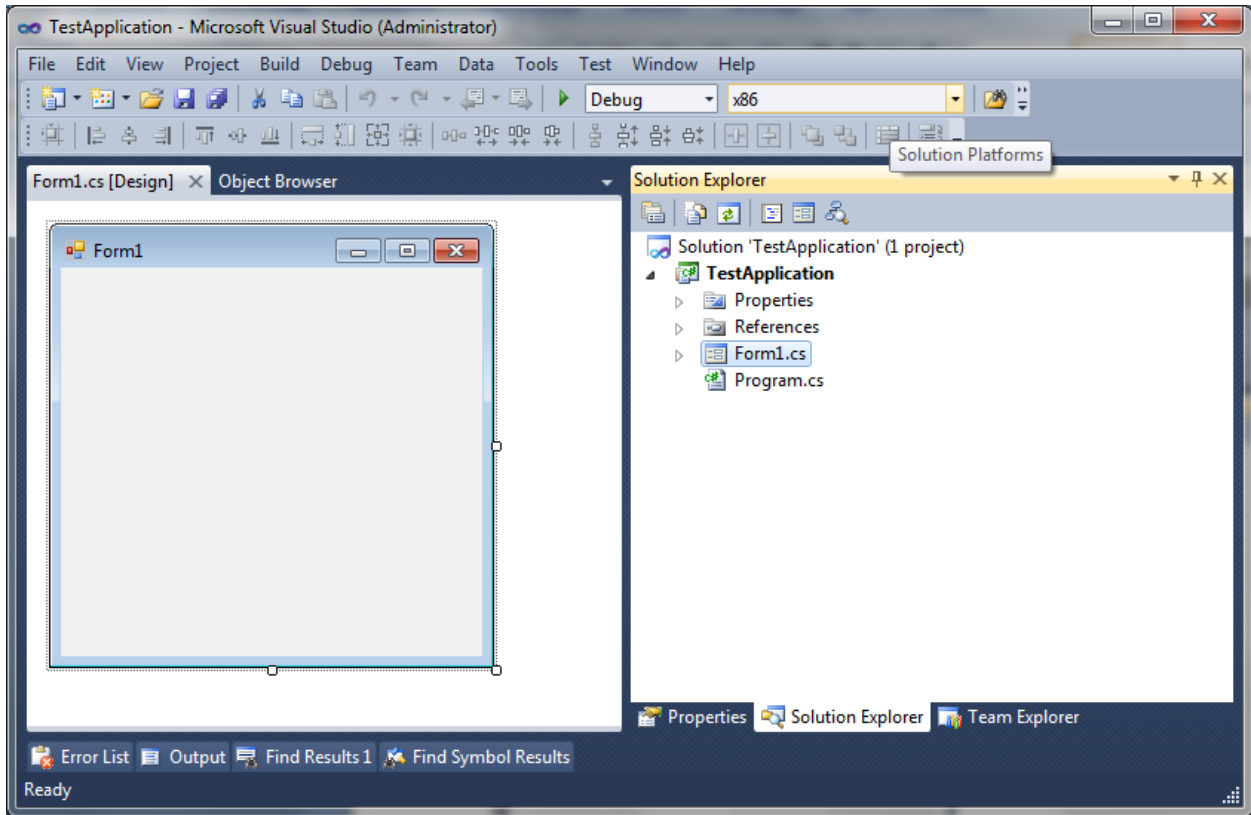


**Figure 13: New Project Window**

Choose a Visual C# Project. In our sample, we will choose a Windows Forms application. Then, select a name for your project, choose a location, and click the **OK** button to confirm.

## 1.2. Step2: Select the Target .NET Framework

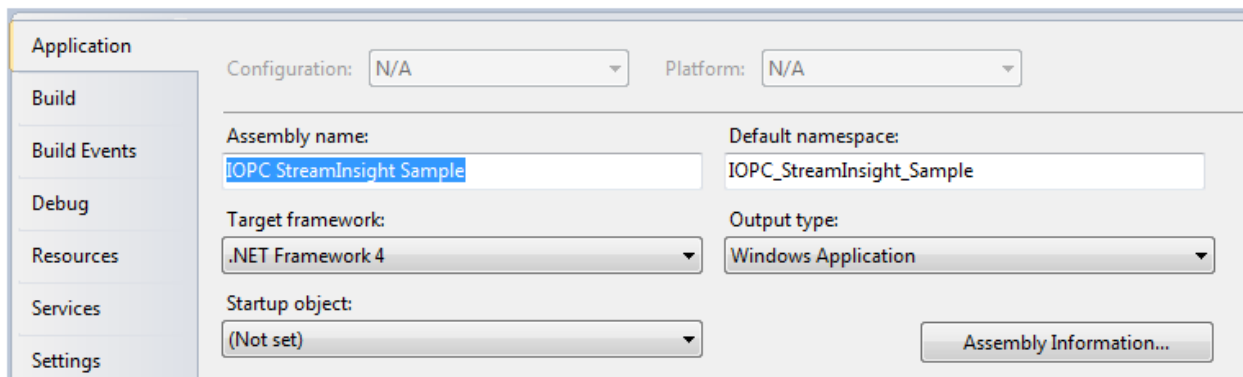
A project named “TestApplication” will be automatically created with a form called “Form1”.



**Figure 14: Project Explorer Window**

Use .NET Framework version 3.5 as the application’s target framework or .NET Framework version 4 if you want to use the WCF debugger tool.

To do so, please go to “Project” → “Settings” and select the “Application” Tab. Select “Target framework” option and set it to the .Net Framework 3.5 or 4 if you will use the WCF Debugger tool or StreamInsight 2.0 or StreamInsight 2.1 or StreamInsight 2.3 (See Figure 15).



**Figure 15: Select .NET framework**

### 1.3. Step3: Add the Required References

To be able to implement a StreamInsight application and use the OPC Adapter for Microsoft StreamInsight, you need to add the following references:

- **Microsoft StreamInsight SDKs** which are:
  - Microsoft.ComplexEventProcessing.dll
  - Microsoft.ComplexEventProcessing.Adapters.dll
  - Microsoft.ComplexEventProcessing.ManagementService.dll

Typically, these dlls are available in the folder: .\Program Files\Microsoft StreamInsight 1.2\Bin

- **OPC Adapter for Microsoft StreamInsights** which are:
  - IntegrationObjects.OPCDA.InputAdapter.SDK.dll : This is the Integration Objects' OPC StreamInsight input adapter.
  - IntegrationObjects.OPCDA.OutputAdapter.SDK.dll : This is the Integration Objects' StreamInsight output adapter.

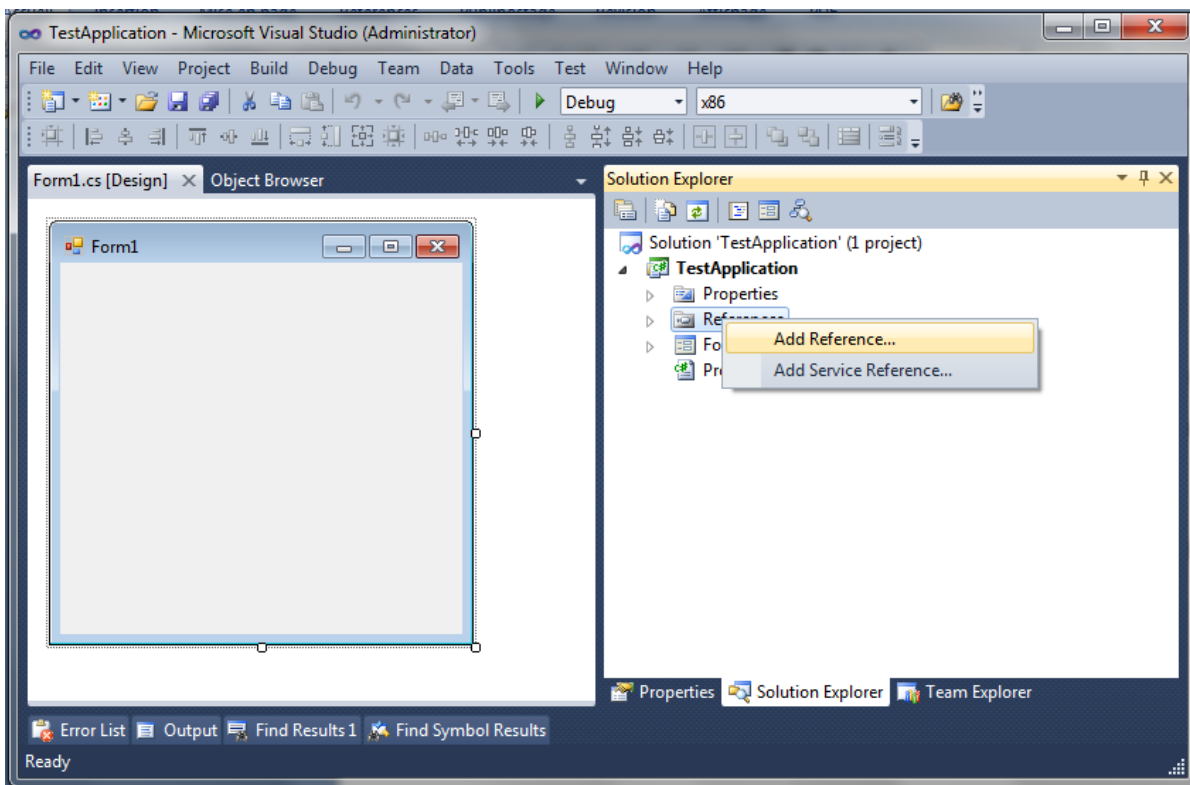
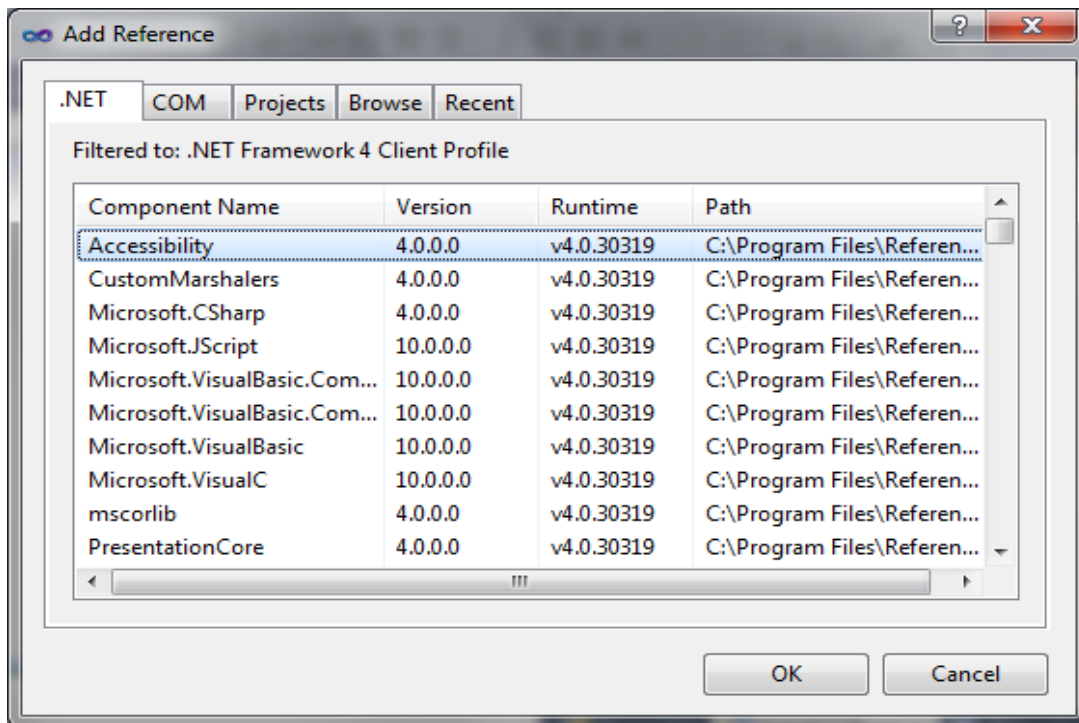
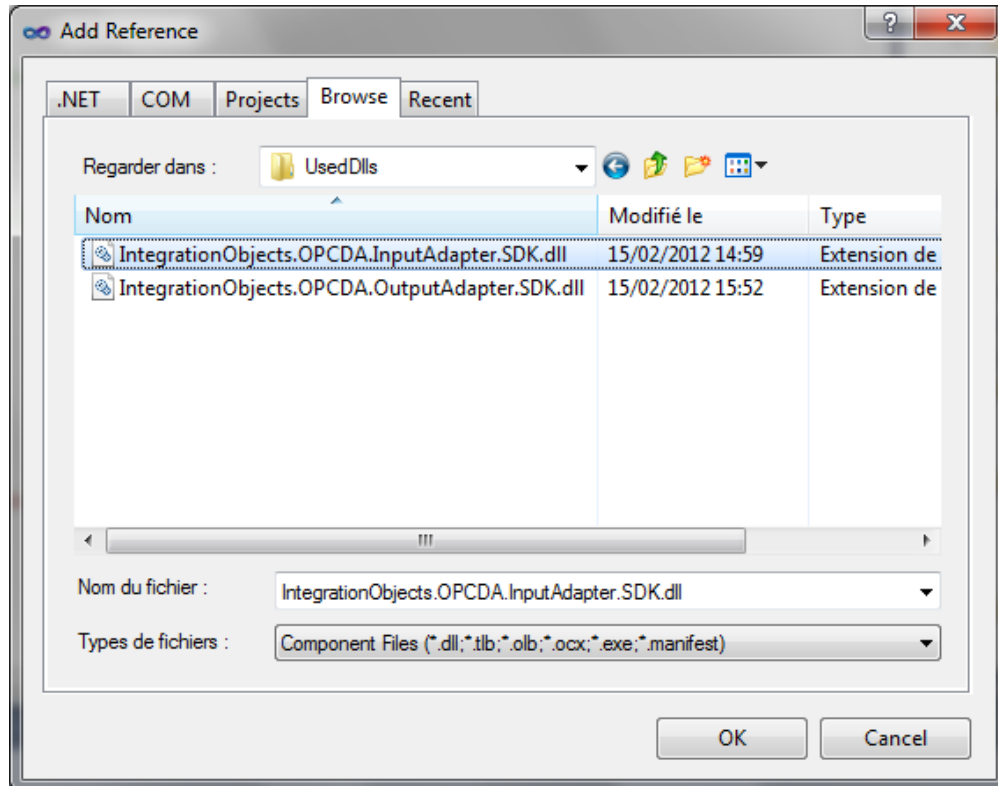


Figure 16: Add Reference Menu





**Figure 17: Adding Reference Dialog**



**Figure 18: OPC Adapter for Microsoft StreamInsight References**

Make sure to copy the “license.dll” and “IntegrationObjects.Logger.SDK.dll” files in your output folder. These dlls are core components used by the adapters’ toolkits.

## 2. Implementing a StreamInsight Application

In this section, we will demonstrate how you can build a simple StreamInsight application that uses Integration Objects’ OPC Adapter for Microsoft StreamInsights. You can use the application to analyze your OPC data through a simple query that streams data from one OPC point to another.

### 2.1. Create the User Interface

Create a similar user interface that mainly includes the following graphical components:

- Configuration menu that will contain the following menu items:
  - Generate Config File: This button will display a dialog box to generate CSV configuration files by selecting OPC Servers and Items.
  - Application: This button will open the application configuration window.
  - Input adapter: That will display the input adapter configuration window.
  - Output adapter: That will display the output adapter configuration window.
- Start button: Using this button, the user will start streaming the OPC real-time data in and out of StreamInsight Engine.
- Stop button: this button will stop the streaming of OPC real-time data.
- A data grid view to display to OPC real-time output data from the StreamInsight Engine.

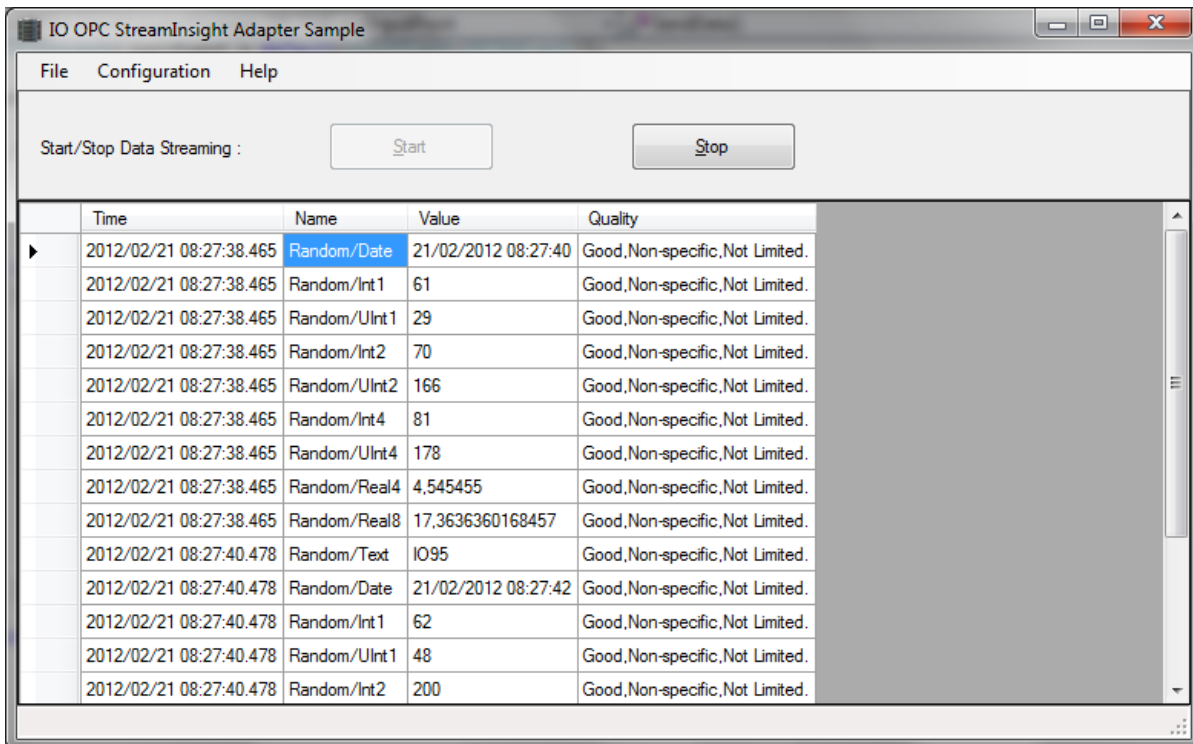


Figure 19: Graphical User Interface

## 2.2. Add Using Directives, Parameters and Main Class

Include the following “Using” directives in the main class:

```
using IntegrationObjects.OPCDA.OutputAdapter;  
using IntegrationObjects.OPCDA.InputAdapter;  
using Microsoft.ComplexEventProcessing;  
using Microsoft.ComplexEventProcessing.Linq;  
using Microsoft.ComplexEventProcessing.Extensibility;  
using Microsoft.ComplexEventProcessing.ManagementService;
```

Add the following parameters to be used in the project later:

```
// The StreamInsight service instance name  
private const string _instanceName = "StreamInsightInstance";  
  
// The StreamInsight application name  
private const string _applicationName = "OPCSITestApplication";  
  
// The OPC StreamInsight input Adapter name  
private const string _inputAdapterName = "OPCInputAdapter";  
// The OPC StreamInsight input Adapter description  
private const string _inputAdapterDescription = "This is the IO OPC SI input Adapter";  
  
// The OPC StreamInsight output Adapter name  
private const string _outputAdapterName = "OPCOutputAdapter";  
// The OPC StreamInsight output Adapter description  
private const string _outputAdapterDescription = "This is the IO OPC SI output Adapter";  
  
// The OPC StreamInsight CTI event frequency  
private const int _input_ctiFrequency=1,  
// The OPC StreamInsight csv file configuration path  
private const string ConfigurationPath=@"c:\OPCTags.csv";  
// The OPC StreamInsight input adapter ticks  
private const int _input_ticks =0;  
// The number reconnection attemptation in case of network trouble shout  
private const int _input_reconnectionTime=5;  
// The StreamInsight input adapter maximum number of trying to send an event  
private const int _input_maxSendTries=0;  
// The StreamInsight input Adapter Production Rate  
private const int _input_ProductionEventRate=1;
```

The process starts with the creation of a StreamInsight server instance and application.

```
static void Main(string[] args)
{
    Microsoft.ComplexEventProcessing.Application _application = null;

    // Create the CEP server in-process.
    //
    // In order to connect to an existing server instead of creating a
    // new one, Server.Connect() can be used, with the appropriate
    // endpoint exposed by the remote server

    using (Server cepServer = Microsoft.ComplexEventProcessing.Server.Create(_instanceName))
    {
        // Create application in server. The application will serve
        // as a container for actual CEP objects and queries.

        _application = cepServer.CreateApplication(_applicationName);
        // ...
    }
}
```

A server must be created with an instance name that has been registered on the machine during the StreamInsight setup process.

In our example, we use a StreamInsight server instance (`_instanceName`) that was declared in the previous constant parameters. An alternative deployment scenario is to use a StreamInsight service on the local or remote server.

The Application (“`_application`” instance) will serve as a container for the actual CEP objects and queries.

To get a list of installed StreamInsight service instances in the local machine, the user can use the “`GetStreamInsightInstances`” method of the OPC input adapter as shown below:

```
// Getting the list of StreamInsight instances names that are contained in the current
machine
List<string> listInstance = StreamInsightManager.GetStreamInsightInstances(out strError);
```

### 2.3. Configure OPC Adapter for Microsoft StreamInsights

OPC StreamInsight input adapter retrieves data from OPC Server into the StreamInsight server engine and the output adapter streams the events from the StreamInsight server engine to a client application. OPC StreamInsight input and output adapters are configured independently.

## 2.4. Configure OPC StreamInsight Input Adapter

The input adapter configuration specifies how the OPC real-time data will be streamed. First, create an input configuration object as shown in the sample code below:

```
//Create an input configuration object

    OPCDAEventInputConfig inputConfiguration= new OPCDAEventInputConfig
    {
        CtiFrequency = _input_ctiFrequency,
        ConfigurationPath = _input_configurationFilePath,
        Ticks = _input_ticks,
        ReconnectionTime = _input_reconnectionTime,
        FiablityLevel = _input_fiablilyLevel,
        dataProductionRate = _input_ProductionEventRate,
    };
```

Next, an input stream is created on top of the existing adapter implementation.

```
// The OPC InputStream Name
string strStream1 = "Stream_" + DateTime.Now.Ticks;
// Create an OPC DA Event stream
Microsoft.ComplexEventProcessing.Linq.CepStream<OPCDAEvent> stream1 =
CepStream<OPCDAEvent>.Create(strStream1);
```

This creates a CepStream object that represents an event stream and is produced, once the query is started, by an adapter instantiated through the factory class. The stream is given a name that can be used later to retrieve stream-specific diagnostics.

Now, we have to create the input adapter factory model as the following:

```
// Create an input Adapter Factory Model

    InputAdapter inputAdapterOPCDA =
_application.CreateInputAdapter<OPCDAEventInputFactory>(_input_Name, "OPC StreamInsight
Input");
```

## 2.5. Define the Query

The input stream (the CepStream object) is used as the basis for the query definition. The query will run continuously to process events received from the input stream it is bound to and will send processed events to the output adapter to which it is bound to.

The CepStream object (stream1) is used as the basis for the definition of the actual query logic. The query uses LINQ as the query specification language:

```
// Create a simple Query by selecting all the values without Filtering
var queryOutput = from e in stream1
                  select e;
```

Create the query template and the query binder as shown below:

```
//Create the query template
Microsoft.ComplexEventProcessing.QueryTemplate template =
_application.CreateQueryTemplate("test", "test", queryOutput);

// bind the query template
Microsoft.ComplexEventProcessing.QueryBinder binder = new QueryBinder(template);
```

In the next step, we have to specify the input adapter factory, the configuration object and the input stream. In this example, we will use point as shown below:

```
// Corresponding the input Adapter to configuration object and the stream
binder.BindProducer<OPCDAEvent>(strStream1, inputAdapterOPCDA, _inputConfig1,
EventShape.Point);
```

## 2.6. Configure OPC StreamInsight Output Adapter

The output adapter streams the OPC real-time data to a client application.

Similarly to the input stream, the output adapter requires the specification of an output adapter factory, a configuration object, an output stream shape and temporal ordering.

In the following code example, you can find how to add an output configuration:

```
// Creating an output configuration
OutputAdapterConfiguration outputConfigObject = new OutputAdapterConfiguration();

// subscribe the data change event
outputConfigObject.SubscribeToDataChangeEvent().DataChange += new
DataEvent.DataChangeEventHandler(Tester_DataChange);
```

The output configuration provides an event that the client application should implement to receive the OPC data changes.

Below is an example for creating the output adapter model:

```
//Create an output Adapter Model and register it
OutputAdapter ouputAdapterEvent =
_application.CreateOutputAdapter<OPCDAEventOutputFactory>(_output_Name, "OPC StreamInsight
output");
```

Now, you need to specify the output adapter to configuration object and query result:

```
// Corresponding the output Adapter to configuration object and the query result
binder.AddConsumer("queryresult", ouputAdapterEvent, outputConfigObject, EventShape.Point,
StreamEventOrder.FullyOrdered);
```

The event shape defines the temporal characteristics (shape) of the event at the query output. There are three types of event shape as described below:

1. **EventShape.Point:** Any result event lifetime is reduced to a point event.
2. **EventShape.Interval:** Any result event is interpreted as interval event. It is only outputted if its full lifetime is committed by a Current Time Increment (CTI) event.
3. **EventShape.Edge:** Any result event will be interpreted as edge event. Its start time is outputted as a start edge, and its end time as the corresponding end edge.

The **stream event order** parameter affects liveliness of interval event output streams. FullyOrdered means that interval events are always output in the order of their start times, while ChainOrdered produces an output sequence that is ordered by the interval end times.

## 2.7. Creating the CEPQuery

Now, we need to create a query instance that can be started. This allows the stream to be turned into that query. The created query is as shown in the following example:

```
Microsoft.ComplexEventProcessing.Query cepQuery = _application.CreateQuery("Query", "",  
binder);  
//binder parameter: the query binder object created previously.
```

## 2.8. Starting the Query

The last step is to start the query. In this example, the query is stopped when the user clicks the "Stop" button in the Main Interface.

```
//Start the query and wait for the output adapter to signal  
// that it has finished receiving events.  
cepQuery.Start();
```

The data will be displayed in the created datagridview. We will receive the data in the dataChangeEvent that we previously subscribed to as shown below:

```
// subscribe the data change event  
outputConfigObject.SubscribeToDataChangeEvent().DataChange += new  
DataEvent.DataChangeEventHandler(Tester_DataChange);
```



The method “Tester\_DataChange” will receive the data from the output adapter and display it in the datagridview as shown in the following code:

```

foreach (object data in dataValues)
    {
        string[] test = new string[4];
        test[0] = ((DataEntity)data).TimeStamp.ToString("yyyy/MM/dd
HH:mm:ss.fff");
        test[1] = ((DataEntity)data).ItemName;
        test[2] = ((DataEntity)data).Value.ToString();
        test[3] = ((DataEntity)data).Quality + "";
        AddEventOPCDataGrid(test);
    }

/// <summary>
/// delegate to use to add a row to the Datagridview
/// </summary>
/// <param name="RowToDisplay">The rows to display</param>
private delegate void CallAddEventToDisplayOPCData(string[] RowToDisplay);

/// <summary>
/// Adds data to the datagridView
/// </summary>
/// <param name="RowToDisplay">rows to display</param>

public void AddEventOPCDataGrid(string[] RowToDisplay)
{
    if (opcDataGrid.InvokeRequired)
    {
        CallAddEventToDisplayOPCData callback = new
CallAddEventToDisplayOPCData(AddEventOPCDataGrid);
        opcDataGrid.BeginInvoke(callback, new object[] { RowToDisplay });
    }
    else
    {
        // Building the Grid Header
        if (opcDataGrid.Columns.Count == 0)
        {
            opcDataGrid.Columns.Add("Time", "Time");
            opcDataGrid.Columns.Add("Name", "Name");
            opcDataGrid.Columns.Add("Value", "Value");
            opcDataGrid.Columns.Add("Quality", "Quality");
        }
        else
        {
            // Displaying the data change values into the dataGrid element
            opcDataGrid.Rows.Add(RowToDisplay);

            if (opcDataGrid.Rows.Count > _output_maxRows)
            {
                opcDataGrid.Rows.RemoveAt(0);
            }
        }
    }
}

```

## 2.9. Stopping the Query

To stop the query, use the following source code:

```
// stopping the Query  
cepQuery.Stop();
```

# USING IO STREAMINSIGHT SAMPLES

## 1. Overview

This section demonstrates how to use and configure the OPC Adapter for Microsoft StreamInsight samples included in the setup package. There are two samples:

- **The Windows Forms sample:** in which the streamed data from the output adapter will be displayed in a data grid view. The user can configure the number of rows displayed in the dataGridView.
- **The console sample:** in which all the streamed data from the output adapter will be displayed in a console.

## 2. OPC StreamInsight WinForms Sample

### 2.1. Open the Sample with Visual Studio2010

Open the OPC Adapter for Microsoft StreamInsight Windows Forms sample located under the following path:

**.\Integration Objects\Integration Objects' OPC Adapter for Microsoft StreamInsight\Samples\OPC StreamInsight WinForms sample Project**

The project will be loaded as shown below:

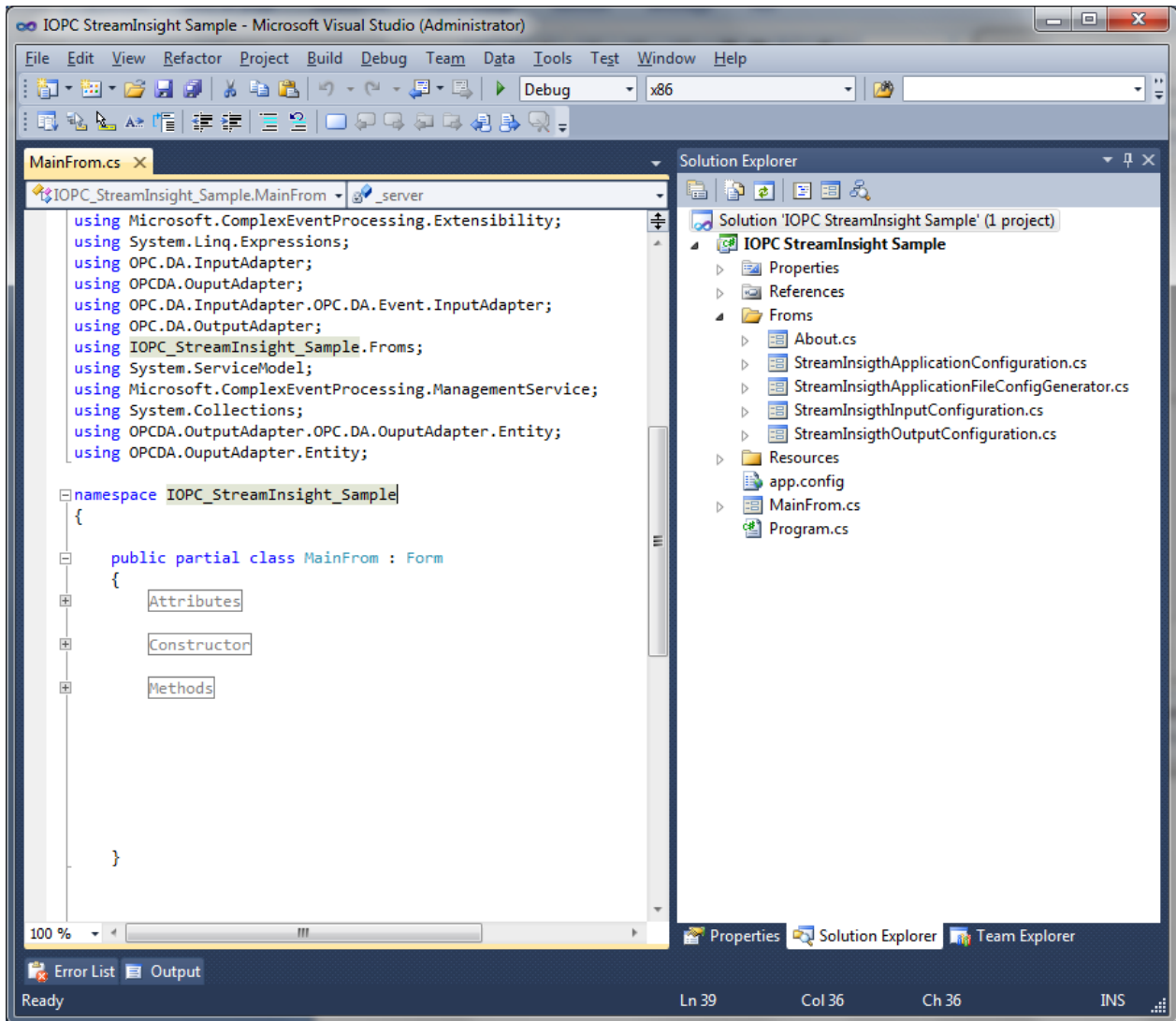
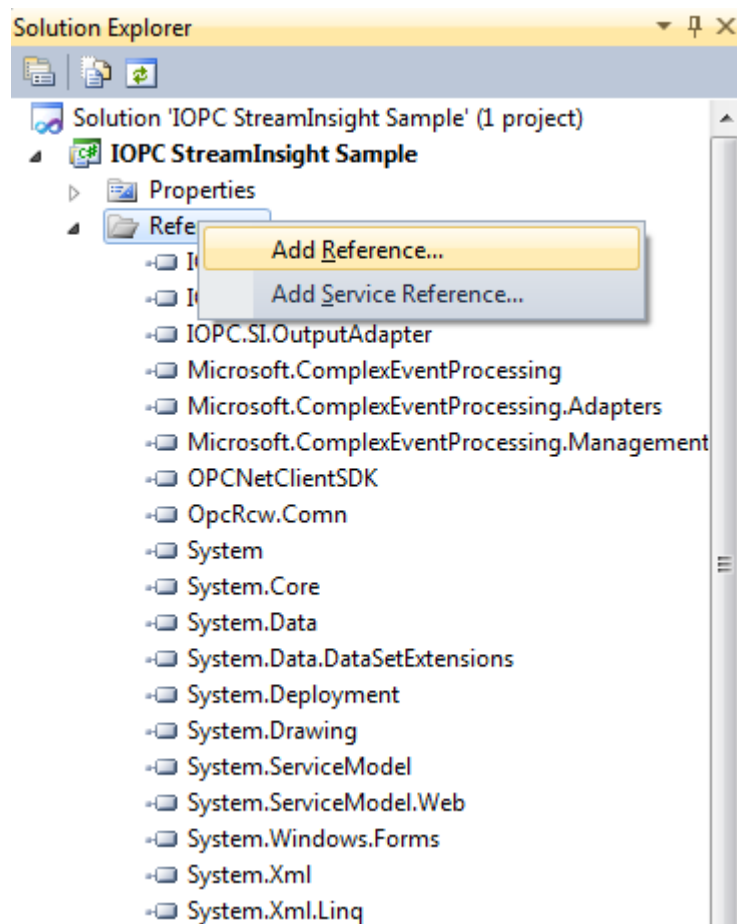


Figure 20: Project Explorer

## 2.2. Add the Required References

Under the project explorer, go to References → Right Click → Add Reference as shown below:



**Figure 21: Add Reference**

Add the following references:

- **IntegrationObjects.OPCDA.InputAdapter.SDK.dll**
- **IntegrationObjects.OPCDA.OutputAdapter.SDK.dll**

You can find these dlls under the following path:

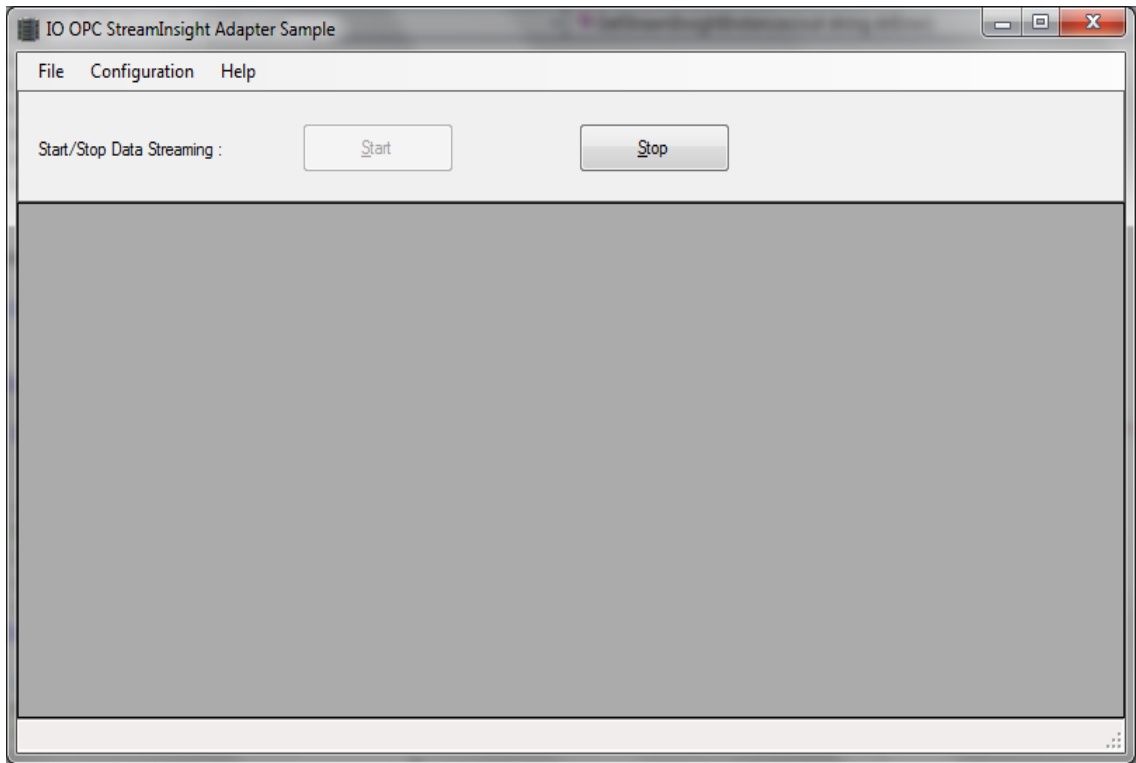
**.\ Integration Objects\Integration Objects' OPC Adapter for Microsoft StreamInsight\Bin**

Also, make sure that these files are under your output folder:

- **IntegrationObjects.Logger.SDK.dll**
- **OPCRcw.Comn.dll**
- **License.dll**

### 2.3. Generate CSV File

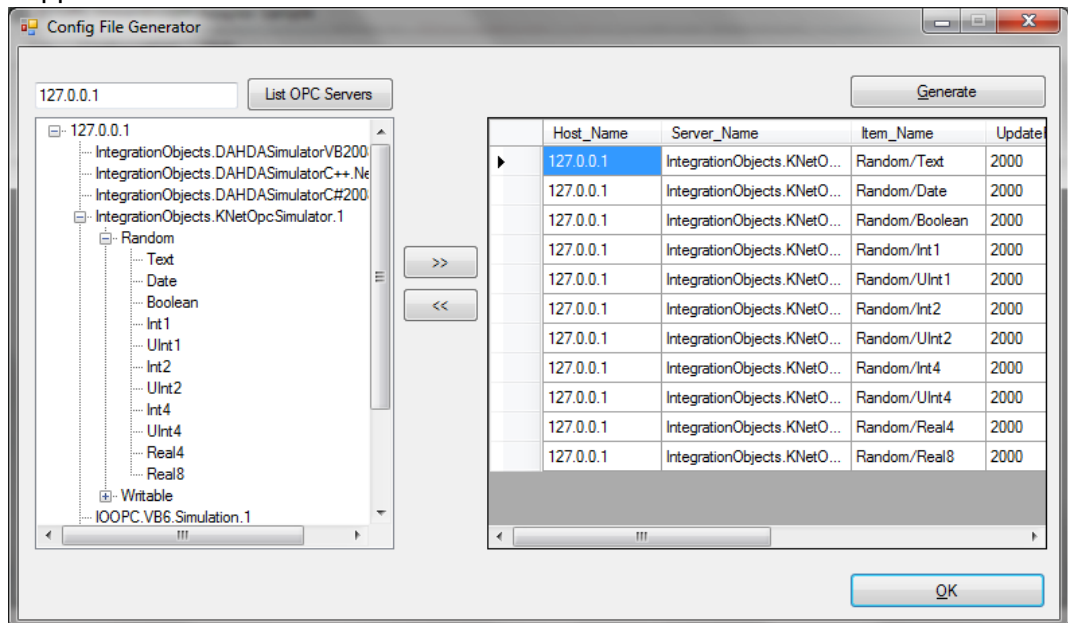
Once all your dlls are referenced correctly, run the project, the following window will be displayed:




**Figure 22: Main Dialog of the sample**

You first need to configure the OPC Servers and Items by generating a configuration CSV file that will be the input to OPC StreamInsight input adapter. To do so, proceed as follow:

1. Go to “Configuration” → “Generate Config files”, the following window will appear:



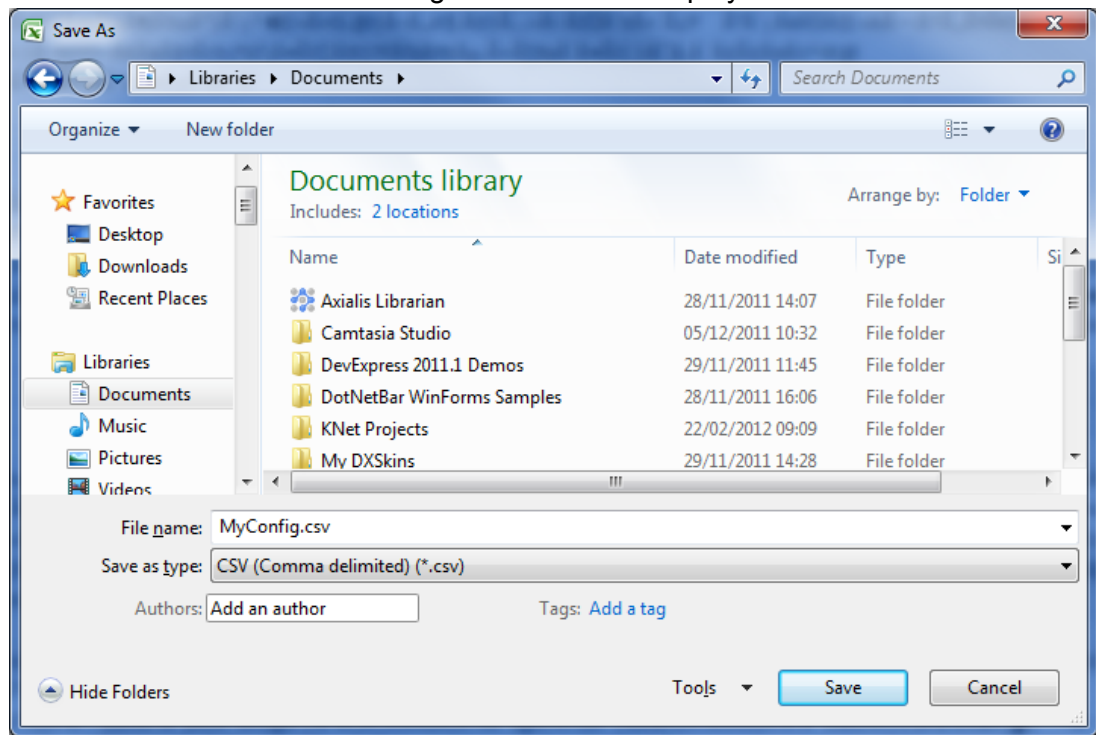
**Figure 23: Configuration File Generator Window**

2. Click the “List OPC Servers” button to get the list of available OPC Servers.
3. In the tree View of OPC Servers, double click on the OPC Server that you want to use in you configuration, browse its address space and click on the following button  to add the selected OPC Items.



**Note:** From the data grid view at the right side, you can modify the update rate, the read mode and the alias name of the OPC Item.

4. Once you finish adding the OPC Items that you want to supervise, click the “Generate” button. The following window will be displayed :



**Figure 24: Select folder Dialog**

5. Select the folder where you want to save the generated CSV file and click the “Save” button. The generated CSV file will be saved at the specified location.
6. Now click the “OK” button of the config Generator dialog.

The generated CSV file is as shown below:

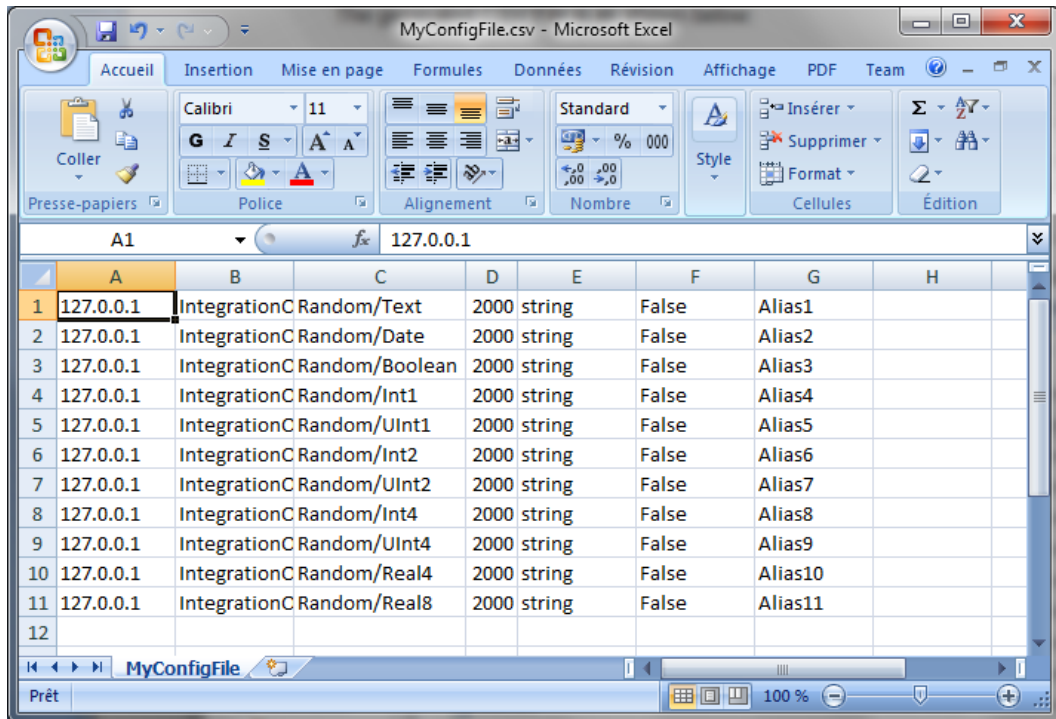


Figure 25: Example of Configuration File

## 2.4. Configure the Input Adapter

To configure the input adapter, go to “Configuration”→ “Input Adapter”. The following dialog will be opened:

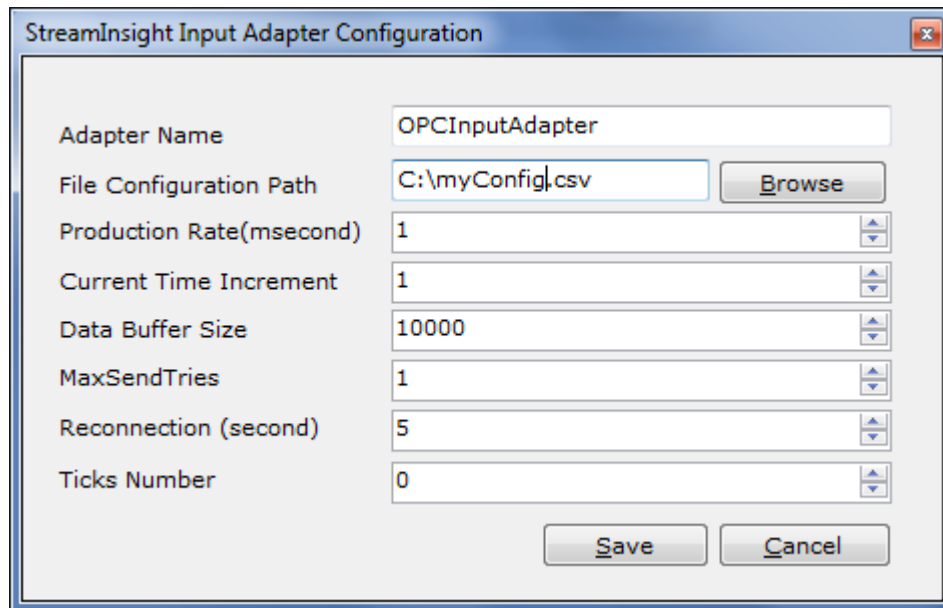


Figure 26: Input Adapter Configuration Window

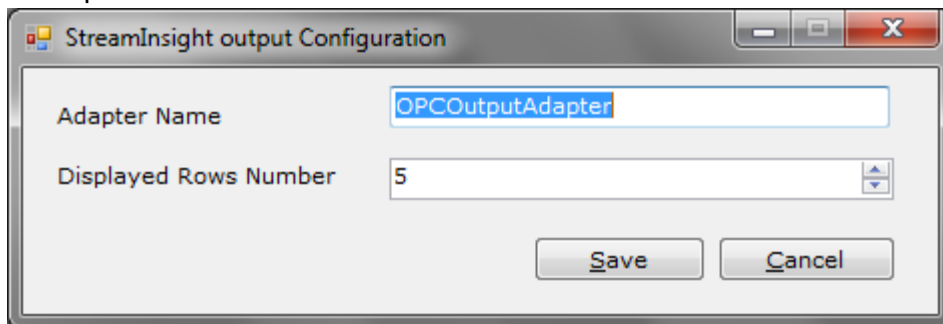


Enter the required fields:

- **Adapter Name:** Give a name to your input adapter. The default displayed name is “OPCInputAdapter”.
- **File Configuration Path:** The path of the CSV file configuration generated in the previous step.
- **Event Production (msecond):** Data events production rate in milliseconds.
- **Current Time Increment:** The current time increment which is used to sort the traffic.
- **Data Buffer Size:** The size of the buffer that will hold the data to be streamed into StreamInsight. It is recommended to size the data buffer depending on the specifications of your machine.
- **MaxSendTries:** The number of attempts the adapter will try to send an event before the operation would be considered as failed.
- **Reconnection (second):** The period of time (in seconds) after which the input adapter will try to reconnect to an OPC Server if the connection was lost.
- **Ticks Number:** The duration in milliseconds to start the event production.

## 2.5. Configure the Output Adapter

To configure the output adapter, go to “Configuration” → “Output Adapter”. The following dialog will be opened:



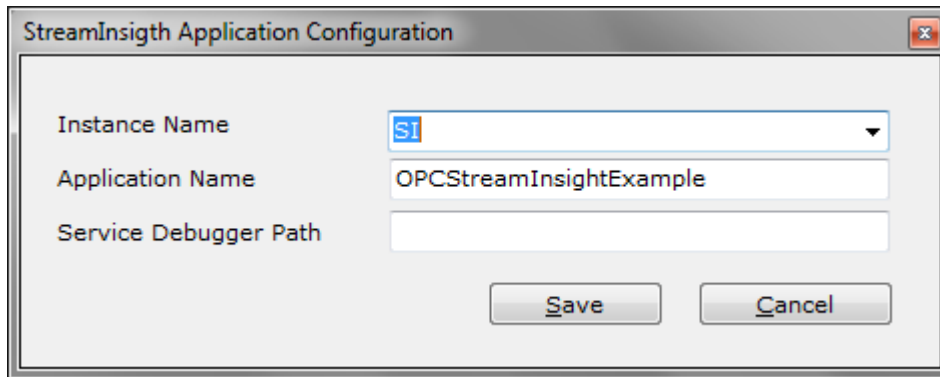
**Figure 27: Output Adapter Configuration Window**

Specify the required fields:

- **Adapter Name:** Give a name to your output adapter. The default displayed name is “OPCOutputAdapter”.
- **Displayed Rows Number:** The number of rows that will be added to the DataGridView and that will display the received data.

## 2.6. Configure Instance and Application Names

Go to “Configuration” → “Application” → select the “Properties” menu button. The following window will be displayed:



**Figure 28: StreamInsight Application Configuration**

This window allows you to configure the following parameters:

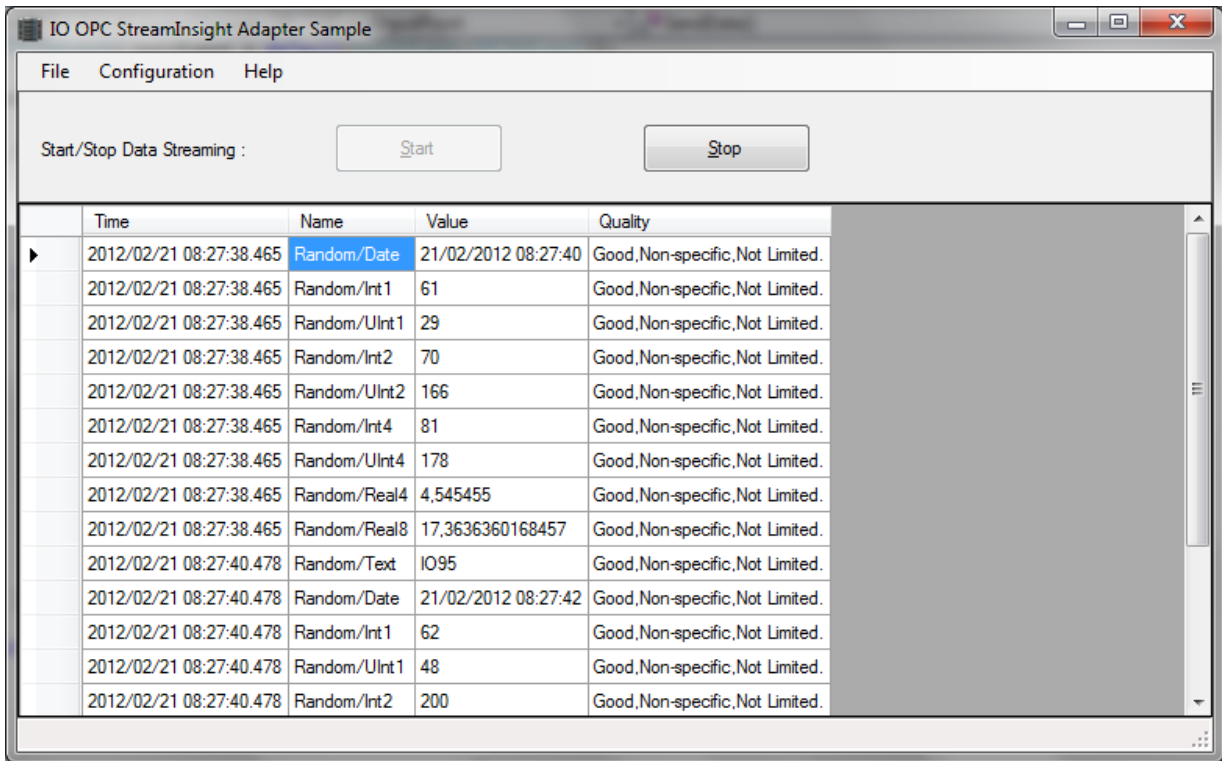
- **Instance Name:** The StreamInsight instance name. By default, this value is the StreamInsight instance installed in your machine.
- **Application Name:** A new of the application.
- **Service Debugger Path:** The path to WCF debugger tool.

Once all the information is entered, click the “Save” button to save the configuration.

## 2.7. Start the StreamInsight Application

After configuring the input and output adapters and specifying the StreamInsight instance and the application names, you can start your sample by clicking the “Start” in the main window.

Your OPC real-time data will be streamed into and out of the StreamInsight Engine and be displayed in the main window as shown below:



The screenshot shows a window titled "IO OPC StreamInsight Adapter Sample" with a menu bar (File, Configuration, Help) and two buttons: "Start" and "Stop". Below the buttons is a table displaying real-time data. The table has five columns: Time, Name, Value, and Quality. The data is as follows:

Time	Name	Value	Quality
2012/02/21 08:27:38.465	Random/Date	21/02/2012 08:27:40	Good,Non-specific,Not Limited.
2012/02/21 08:27:38.465	Random/Int1	61	Good,Non-specific,Not Limited.
2012/02/21 08:27:38.465	Random/UInt1	29	Good,Non-specific,Not Limited.
2012/02/21 08:27:38.465	Random/Int2	70	Good,Non-specific,Not Limited.
2012/02/21 08:27:38.465	Random/UInt2	166	Good,Non-specific,Not Limited.
2012/02/21 08:27:38.465	Random/Int4	81	Good,Non-specific,Not Limited.
2012/02/21 08:27:38.465	Random/UInt4	178	Good,Non-specific,Not Limited.
2012/02/21 08:27:38.465	Random/Real4	4,545455	Good,Non-specific,Not Limited.
2012/02/21 08:27:38.465	Random/Real8	17,3636360168457	Good,Non-specific,Not Limited.
2012/02/21 08:27:40.478	Random/Text	IO95	Good,Non-specific,Not Limited.
2012/02/21 08:27:40.478	Random/Date	21/02/2012 08:27:42	Good,Non-specific,Not Limited.
2012/02/21 08:27:40.478	Random/Int1	62	Good,Non-specific,Not Limited.
2012/02/21 08:27:40.478	Random/UInt1	48	Good,Non-specific,Not Limited.
2012/02/21 08:27:40.478	Random/Int2	200	Good,Non-specific,Not Limited.

Figure 29: Displayed OPC real-time data

### 2.8. Stop the StreamInsight Application

To stop streaming data in/out of StreamInsight Engine, click the “Stop” button in the main interface.

## 3. OPC StreamInsight Console Sample

To use the OPC StreamInsight console sample, you first need to open the project with Visual Studio 2010 as described in the sections 1.1 and 1.2.

Once the environment is set correctly, you can start your application. The following window will be displayed:

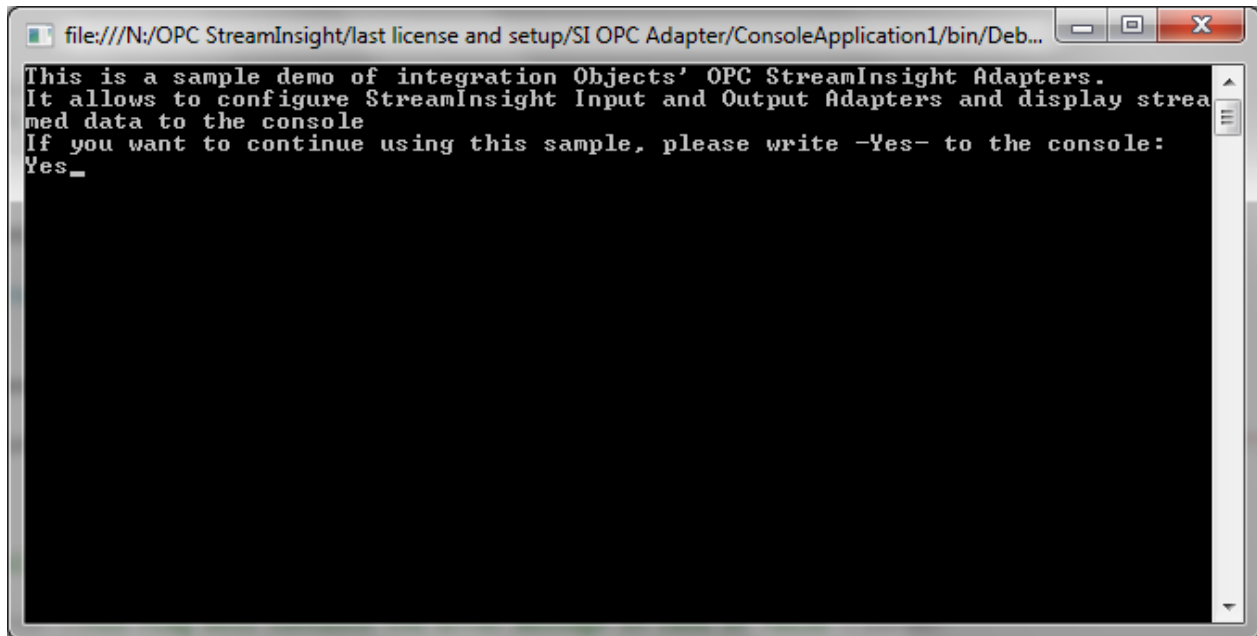
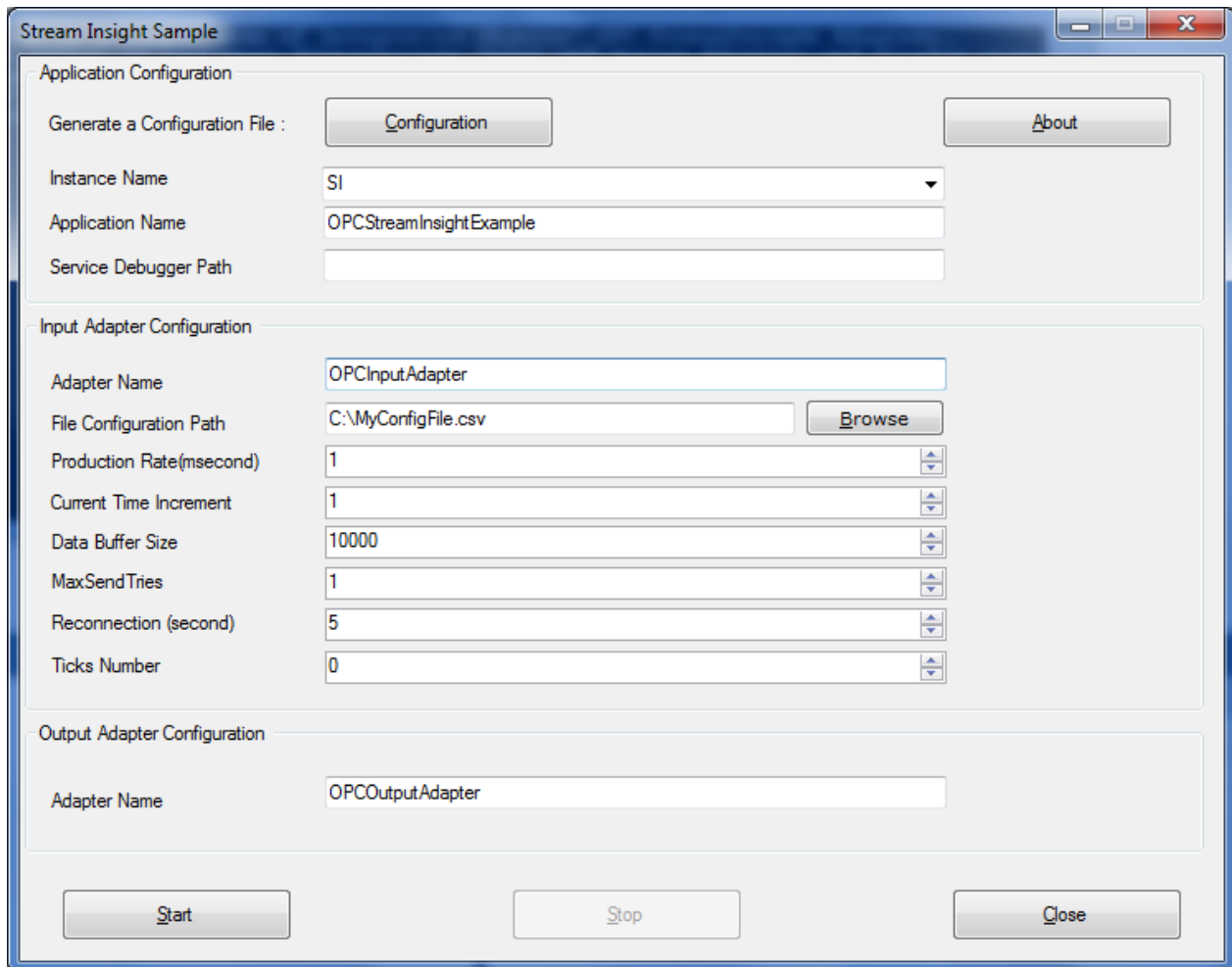


Figure 30: OPC StreamInsight Console Sample

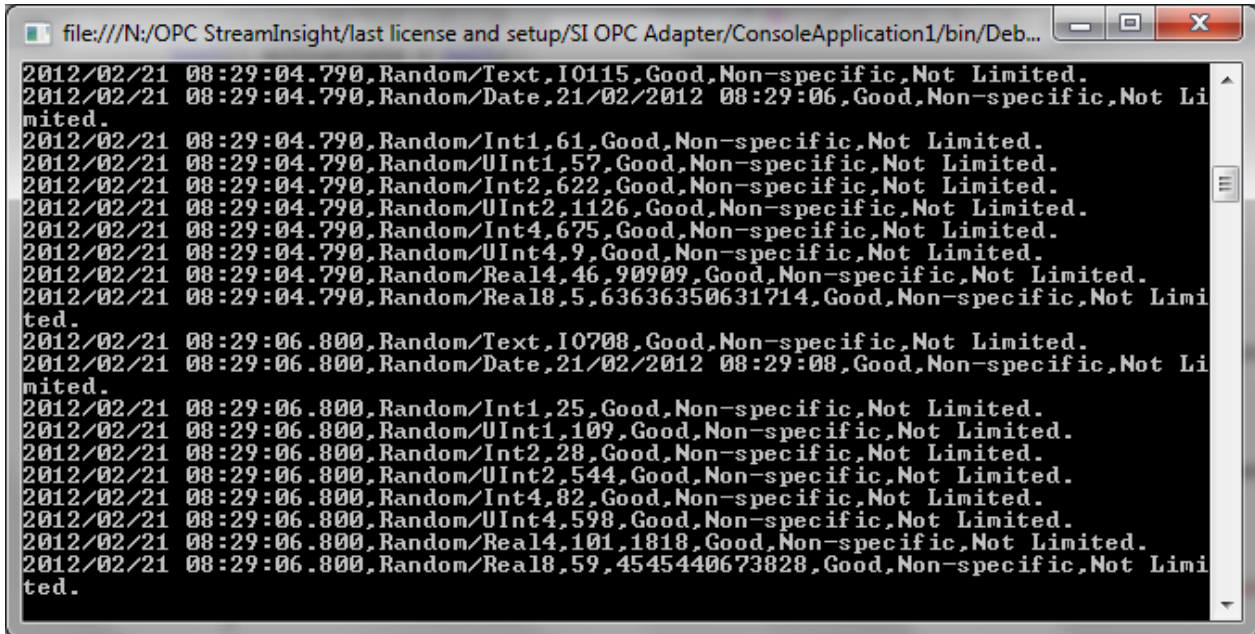
Follow the instructions displayed in the console and you will get the following dialog box:



**Figure 31: Configuration Dialog Box**

In this dialog, you will need to configure the application, the input and output adapters as described in the sections 1.3 → 1.6.

Once the configuration is completed, you can start StreamInsight send/receive events process by clicking the “Start” button. Received OPC real-time data will be displayed on the console as shown below:



```

2012/02/21 08:29:04.790,Random/Text,I0115,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/Date,21/02/2012 08:29:06,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/Int1,61,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/UInt1,57,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/Int2,622,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/UInt2,1126,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/Int4,675,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/UInt4,9,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/Real4,46,90909,Good,Non-specific,Not Limited.
2012/02/21 08:29:04.790,Random/Real8,5,63636350631714,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/Text,I0708,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/Date,21/02/2012 08:29:08,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/Int1,25,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/UInt1,109,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/Int2,28,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/UInt2,544,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/Int4,82,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/UInt4,598,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/Real4,101,1818,Good,Non-specific,Not Limited.
2012/02/21 08:29:06.800,Random/Real8,59,4545440673828,Good,Non-specific,Not Limited.
  
```

Figure 32 : Displaying Data in Console

## 4. How to Use the OPC Adapters with StreamInsight 2.0

To use the OPC SI Input and Output adapters with the version 2.0 of StreamInsight, you need to verify the following items:

- a. Check if .NET Framework 4.0 is installed.
- b. Add the code below to the “app.config” file of the sample that you want to run:

```

<configuration>
<runtime>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
<assemblyIdentity name="Microsoft.ComplexEventProcessing"
publicKeyToken="89845dcd8080cc91"/>
  <bindingRedirect oldVersion="12.0.0.0" newVersion="20.0.0.0"/>
  </dependentAssembly>
</assemblyBinding>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
<dependentAssembly>
  <assemblyIdentity name="Microsoft.ComplexEventProcessing.Adapters"
publicKeyToken="89845dcd8080cc91"/>
  <bindingRedirect oldVersion="12.0.0.0" newVersion="20.0.0.0"/>
  </dependentAssembly>
  </assemblyBinding>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity
name="Microsoft.ComplexEventProcessing.ManagementService"
publicKeyToken="89845dcd8080cc91"/>
    <bindingRedirect oldVersion="12.0.0.0"
newVersion="20.0.0.0"/>
  </dependentAssembly>
  </assemblyBinding>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity
name="Microsoft.ComplexEventProcessing.Observable"
publicKeyToken="89845dcd8080cc91"/>
    <bindingRedirect oldVersion="12.0.0.0"
newVersion="20.0.0.0"/>
  </dependentAssembly>
  </assemblyBinding>
</runtime>
</configuration>

```

c. Recompile the sample project.



If you are using **StreamInsight 2.1** you just need to set the "**newVersion**" to "21.0.0.0".

If you are using **StreamInsight 2.3** you just need to set the "**newVersion**" to "23.0.0.0".

# TRACING CAPABILITIES

## 1. Overview

The adapters' toolkits have several tracing capabilities. They produce 3 log files which are:

- The "StreamInsight\_OPC\_DA\_InputAdapterLogFile.LOG" that records events, errors, and debugging information for the input adapter SDK.
- The "StreamInsight\_OPC\_DA\_OuputAdapterLogFile.LOG" that records events, errors, and debugging information for the output adapter SDK.
- The "StreamInsight\_OPCCliientSDK\_Log.LOG" that records events, errors, and debugging information for OPC related operations.

If difficulties occur with the OPC Adapter for Microsoft StreamInsights, these log files can be extremely valuable for troubleshooting. Under normal operation, the toolkits log very little information.

The Integration Objects' OPC Adapter for Microsoft StreamInsight incorporates 3 configuration files:

- ConfigOPCCliientSDK.ini
- OPCOutputAdapterConfig.ini
- OPCInputAdapterConfig.ini

These files include several logging parameters. These parameters have default settings and can be changed before start-up by editing the configuration files.

To show you how to configure the log settings, we took the OPCInputAdapterConfig.ini file as an example. The configuration of other files is similar.

To change one of the log settings, open OPCInputAdapterConfig.ini in a text editor.

1. Edit any of the parameters listed in the following tables:

Log Setting	Description	Default Value
<b>CreateNew</b>	True to create a new event log or to append the old log.	True
<b>Level</b>	There are five log levels:	Control



	<ol style="list-style-type: none"> <li>1. Control: logs only control messages generated by the input adapter SDK.</li> <li>2. Error: Logs error and control messages generated by input adapter SDK.</li> <li>3. Warning: Logs warning, error, and control messages generated by the input adapter.</li> <li>4. Inform: Logs information, warning, error, and control messages generated by the input adapter SDK.</li> <li>5. Debug: Logs all messages generated by the input adapter SDK.</li> </ol> <p>The higher the log level, the more information is recorded. We recommend using level “Control” for a better performance of the SDK. The other levels are dedicated for troubleshooting purposes.</p>	
<b>Source</b>	The Window event log source name.	StreamInsight OPC DA InputAdapter
<b>LogName</b>	The OPC input adapter SDK log file name.	StreamInsight_InputAdapter_Log
<b>Directory</b>	The folder where the log file will be generated.	OPC DA StreamInsight Log Files
<b>AutoAppend</b>	Set to true to continue writing log messages in the existed log file or to false to create a new file.	True
<b>FileMaxSize</b>	The file maximum size.	10000000 bytes
<b>BufferSize</b>	The maximum number of messages to be stored in the runtime memory before launching a write action in the hard disk. The specified value must be greater than 100.	100
<b>FilePerDay</b>	Set to true to create a new log file every day.	True

**Table 25: Log Settings**

2. Save the file for the log settings for new parameters to take effect.
3. Restart your application.

## 2. Sample Configuration File

```
[LogConfiguration]
CreateNew=True
Level=Debug
LogName= StreamInsight_InputAdapter_Log
Source= StreamInsight OPC DA InputAdapter
Directory=OPC DA StreamInsight Log Files

[FileLogConfiguration]
AutoAppend=True
BufferSize=100
FileMaxSize=10000000
FileName=
StreamInsight_OPC_DA_InputAdapterLogFile
FilePerDay=True
```

# GLOSSARY

## A

### **API**

An application programming interface (API) is a source code-based specification intended to be used as an interface by software components to communicate with each other.

## C

### **COM**

Component Object Model is a specification for writing reusable software components. COM is an infrastructure that allows objects to communicate between processes and computers.

### **CEP**

Complex event processing consists of processing many events happening across all the layers of an organization, identifying the most meaningful events within the event cloud, analyzing their impact, and taking subsequent action in real time.

## O

### **OPC**

OPC is all about Open Productivity & Connectivity in industrial automation and the enterprise systems that support industry. Interoperability is assured through the creation and maintenance of open standards specifications. There are currently seven standards specifications completed or in development.

## P

### **ProgID**

A ProgID or Programmatic Identifier.

The format of a ProgID is <Program>.<Component>.<Version>, separated by periods and with no spaces. Like the CLSID, the ProgID identifies a class but with less precision because it is not guaranteed to be globally unique.

## S

### **SDK**

Software Development Kit is typically a set of software development tools that helps in the creation of software applications.

For additional information on this guide, questions or problems to report, please contact:

**Offices**

- Houston, USA: +1 713 609 9208
- Genova, Italy: +39 34 75 83 93 47
- Tunis, Tunisia: +216 71 195 360

**Email**

- Support Services: [customerservice@integrationobjects.com](mailto:customerservice@integrationobjects.com)
- Sales: [sales@integrationobjects.com](mailto:sales@integrationobjects.com)

To find out how you can benefit from other Integration Objects products and custom-designed solutions, please visit us on the Internet:

**Online**

- [www.integrationobjects.com](http://www.integrationobjects.com)