

# **Integration Objects'**

## **.NET Toolkit for OPC UA Server Applications Development**

**OPC UA Server Toolkit**  
Version 1.0 Rev.0

### **USER GUIDE**

**OPC Compatibility**

OPC Unified Architecture 1.02  
OPC Unified Architecture 1.03

OPC UA Server Toolkit User Guide Version 1.0 Rev.0  
Published January 2020

Copyright © 2018-2020 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®, Windows NT® and .NET are registered trademarks of Microsoft Corporation.

# TABLE OF CONTENTS

PREFACE.....	10
About This User Guide.....	10
Target Audience .....	10
Related Documentation .....	10
Document Conventions.....	10
Customer Support Services .....	11
INTRODUCTION .....	12
1. Overview.....	12
2. Features.....	13
3. Operating Systems Compatibility .....	13
4. OPC Compatibility.....	13
GETTING STARTED .....	14
1. Pre-Installation Considerations .....	14
2. Installation.....	14
3. Uninstallation .....	24
4. Compiling and Linking Applications.....	25
• Step 1: Select your Platform.....	25
• Step 2: Create your Project.....	27
• Step 3: Add your References .....	29
5. Runtime Deployment Procedure .....	30
USING THE OPC UA SERVER TOOLKIT .....	32
1. Initialization of the API.....	32
2. UAServerManager Class .....	32
2.1. Constructor .....	32
2.2. Functions .....	32
2.2.1. Address Space Management .....	32
2.2.1.1. SetDARootFolder.....	32
2.2.1.2. SetHDARootFolder .....	33

2.2.1.3.	SetMethodsRootFolder .....	33
2.2.1.4.	AddVariableToFolder .....	33
2.2.1.5.	AddVariablesToFolder .....	34
2.2.1.6.	AddMethodsToFolder .....	34
2.2.1.7.	AddFolderToFolder .....	34
2.2.1.8.	UpdateDAVariable .....	35
2.2.2.	Start/Stop Server .....	36
2.2.2.1.	StartServer.....	36
2.2.2.2.	StopServer.....	36
2.2.3.	Server Statistics Management .....	38
2.2.3.1.	UpdateSessions.....	38
2.2.3.2.	UpdateSubscriptions.....	38
2.2.3.3.	GetServerStatus .....	38
2.2.3.4.	UpdateEndpoints .....	39
2.2.4.	Server Configuration .....	40
2.2.4.1.	SetIniFilePath .....	40
2.2.4.2.	SetConfigurationFilePath .....	40
2.2.5.	Callbacks .....	41
2.2.5.1.	SubscribeOnSessionCreatedEvent.....	41
2.2.5.2.	UnsubscribeOnSessionCreatedEvent.....	41
2.2.5.3.	SubscribeOnSessionActivatedEvent.....	41
2.2.5.4.	UnsubscribeOnSessionActivatedEvent.....	42
2.2.5.5.	SubscribeOnSessionClosedEvent .....	42
2.2.5.6.	UnsubscribeOnSessionClosedEvent.....	43
2.2.5.7.	SubscribeOnSubscriptionCreatedEvent .....	43
2.2.5.8.	UnsubscribeOnSubscriptionCreatedEvent .....	43
2.2.5.9.	SubscribeOnSubscriptionDeletedEvent .....	44
2.2.5.10.	UnsubscribeOnSubscriptionDeletedEvent.....	44
2.2.5.11.	SubscribeOnWriteValueEvent.....	45
2.2.5.12.	UnsubscribeOnWriteValueEvent.....	45
2.2.5.13.	SubscribeOnReadRawHDAValueEvent .....	45
2.2.5.14.	SubscribeOnReadProcessedHDAValueEvent .....	46

2.2.5.15.	UnsubscribeOnReadProcessedHDAValueEvent.....	46
2.2.5.16.	SubscribeOnReadAtTimeHDAValueEvent .....	46
2.2.5.17.	UnsubscribeOnReadAtTimeHDAValueEvent .....	47
2.2.5.18.	SubscribeOnDeleteAtTimeHDAValueEvent .....	47
2.2.5.19.	UnsubscribeOnDeleteAtTimeHDAValueEvent .....	48
2.2.5.20.	SubscribeOnHistoryUpdateEvent.....	48
2.2.5.21.	UnsubscribeOnHistoryUpdateEvent.....	48
2.2.5.22.	SubscribeOnCertificateValidationEvent.....	49
2.2.5.23.	UnsubscribeOnCertificateValidationEvent.....	49
2.2.5.24.	SubscribeOnHistoryDeleteRaw .....	50
2.2.5.25.	UnsubscribeOnHistoryDeleteRaw .....	50
2.2.6.	Certificate Management .....	51
2.2.6.1.	TrustCertificate .....	51
2.2.6.2.	RejectCertificate.....	51
2.2.7.	Delegates.....	53
2.2.7.1.	SessionsEventsHandler .....	53
2.2.7.2.	SubscriptionsEventsHandler .....	53
2.2.7.3.	WriteValueEventHandler.....	53
2.2.7.4.	ReadRawHDAEventHandler .....	54
2.2.7.5.	ReadProcessedHDAEventHandler.....	55
2.2.7.6.	ReadAtTimeHDAEventHandler .....	55
2.2.7.7.	DeleteAtTimeHDAEventHandler .....	56
2.2.7.8.	CallMethodEventHandler .....	56
2.2.7.9.	CertificateValidatorEventHandler .....	57
2.2.7.10.	HistoryUpdateDataEventHandler .....	57
2.2.7.11.	HistoryDeleteRawEventHandler.....	57
3.	Folder Class.....	59
3.1.	Constructor .....	59
4.	Variable Class.....	59
4.1.	Constructor .....	59
4.2.	Functions .....	60
5.	Method Class.....	61

5.1. Constructor .....	61
5.2. Functions .....	61
5.2.1. ExecuteMethod .....	61
USING OPC UA SAMPLE SERVERS .....	63
1. OPC UA Sample Server C# .....	63
2. OPC UA Sample Service VB.....	66
TOOLKIT TRACING CAPABLITIES.....	68
TROUBLESHOOTING.....	70

## TABLE OF FIGURES

Figure 1: OPC UA Server Toolkit Overview .....	12
Figure 2: Installation Welcome Dialog .....	15
Figure 3: License Agreement Dialog .....	16
Figure 4: Customer Information Dialog .....	17
Figure 5: Setup Type Dialog .....	18
Figure 6: Choose Destination Location Dialog .....	19
Figure 7: Ready to Install the Program Dialog .....	20
Figure 8 : License Activation Dialog .....	21
Figure 9: Installation Completed Dialog .....	22
Figure 10: OPC UA Server Toolkit Start Menu .....	22
Figure 11: OPC UA Sample Server Service .....	23
Figure 12: Uninstaller Shortcut in the Start Menu .....	25
Figure 13: Platform Target for 32-bit Machine .....	26
Figure 14: Platform Target for 64-bit Machine .....	27
Figure 15: New Windows Form Project .....	28
Figure 16: Windows Form Project .....	29
Figure 17: Add Reference .....	29
Figure 18: Choosing a Reference .....	30
Figure 19: Run the OPC UA Sample Server .....	63
Figure 20: OPC UA Sample Server User Interface .....	64
Figure 21: OPC UA Server Endpoints URLs .....	64
Figure 22: Connection Established from an OPC UA Client .....	65
Figure 23: Subscriptions and Monitored Items .....	65
Figure 24: Start the OPC UA Sample Server Service .....	66
Figure 25 : XML Configuration File .....	67
Figure 26: OPC UA Server Toolkit Start Menu .....	70
Figure 27: Select Type Dialog .....	71
Figure 28: License Activation Dialog .....	72

## LIST OF TABLES

Table 1: Files Included in the Distribution .....	24
Table 2 : Parameters of SetDARootFolder .....	32
Table 3: Parameters of SetHDARootFolder .....	33
Table 4: Parameters of SetMethodsRootFolder .....	33
Table 5: Parameters of AddVariableToFolder .....	34
Table 6: Parameters of AddVariablesToFolder .....	34
Table 7: Parameters of AddMethodsToFolder .....	34
Table 8: Parameters of AddFolderToFolder .....	35
Table 9: Parameters of UpdateDAVariable .....	35
Table 10: Returned Results of UpdateDAVariable .....	35
Table 11: Parameters of StartServer .....	36
Table 12: Returned Results of StartServer .....	36
Table 13: Parameters of StopServer .....	36
Table 14: Returned Results of StopServer .....	37
Table 15: Returned Results of UpdateSessions .....	38
Table 16: Returned Results of UpdateSubscriptions .....	38
Table 17: Returned Results of GetServerStatus .....	39
Table 18: Returned Results of UpdateEndpoints .....	39
Table 19: Parameters of SetIniFilePath .....	40
Table 20: Parameters of SetConfigurationFilePath .....	40
Table 21: Parameters of SubscribeOnSessionCreatedEvent .....	41
Table 22: Parameters of UnsubscribeOnSessionCreatedEvent .....	41
Table 23: Parameters of SubscribeOnSessionActivatedEvent .....	42
Table 24: Parameters of UnsubscribeOnSessionActivatedEvent .....	42
Table 25: Parameters of SubscribeOnSessionClosedEvent .....	42
Table 26: Parameters of UnsubscribeOnSessionClosedEvent .....	43
Table 27: Parameters of SubscribeOnSubscriptionCreatedEvent .....	43
Table 28: Parameters of UnsubscribeOnSubscriptionCreatedEvent .....	44
Table 29: Parameters of SubscribeOnSubscriptionDeletedEvent .....	44
Table 30: Parameters of UnsubscribeOnSubscriptionDeletedEvent .....	44
Table 31: Parameters of SubscribeOnWriteValueEvent .....	45
Table 32: Parameters of UnsubscribeOnWriteValueEvent .....	45
Table 33: Parameters of SubscribeOnReadRawHDAValueEvent .....	46
Table 34: Parameters of SubscribeOnReadProcessedHDAValueEvent .....	46
Table 35: Parameters of UnsubscribeOnReadProcessedHDAValueEvent .....	46
Table 36: Parameters of SubscribeOnReadAtTimeHDAValueEvent .....	47
Table 37: Parameters of UnsubscribeOnReadAtTimeHDAValueEvent .....	47
Table 38: Parameters of SubscribeOnDeleteAtTimeHDAValueEvent .....	48
Table 39: Parameters of UnsubscribeOnDeleteAtTimeHDAValueEvent .....	48
Table 40: Parameters of SubscribeOnHistoryUpdateEvent .....	48
Table 41: Parameters of UnsubscribeOnHistoryUpdateEvent .....	49
Table 42: Parameters of SubscribeOnCertificateValidationEvent .....	49
Table 43: Parameters of UnsubscribeOnCertificateValidationEvent .....	50



<b>Table 44: Parameters of SubscribeOnHistoryDeleteRaw</b> .....	50
<b>Table 45: Parameters of UnsubscribeOnHistoryDeleteRaw</b> .....	50
<b>Table 46: Parameters of TrustCertificate</b> .....	51
<b>Table 47: Returned Results of TrustCertificate</b> .....	51
<b>Table 48: Parameters of RejectCertificate</b> .....	52
<b>Table 49: Returned Results of RejectCertificate</b> .....	52
<b>Table 50: Parameters of SessionsEventsHandler</b> .....	53
<b>Table 51: Parameters of SubscriptionsEventsHandler</b> .....	53
<b>Table 52: Parameters of WriteValueEventHandler</b> .....	54
<b>Table 53: Returned Results of WriteValueEventHandler</b> .....	54
<b>Table 54: Parameters of ReadRawHDAEventHandle</b> .....	55
<b>Table 55: Parameters of ReadProcessedHDAEventHandler</b> .....	55
<b>Table 56: Parameters of ReadAtTimeHDAEventHandler</b> .....	56
<b>Table 57: Parameters of DeleteAtTimeHDAEventHandler</b> .....	56
<b>Table 58: Parameters of CallMethodEventHandler</b> .....	56
<b>Table 59: Parameters of CertificateValidatorEventHandler</b> .....	57
<b>Table 60: Parameters of HistoryUpdateDataEventHandler</b> .....	57
<b>Table 61: Parameters of HistoryDeleteRawEventHandler</b> .....	58
<b>Table 62: Parameters of Folder Constructor</b> .....	59
<b>Table 63: Parameters of Variable Constructor</b> .....	59
<b>Table 64: Parameters of UpdateItem</b> .....	60
<b>Table 65: Returned Results of UpdateItem</b> .....	60
<b>Table 66: Parameters of Method Constructor</b> .....	61
<b>Table 67: Parameters of UpdateItem</b> .....	62
<b>Table 68: Log Settings</b> .....	69

# PREFACE

## ABOUT THIS USER GUIDE

This guide describes the functions provided by Integration Objects' OPC UA Server Toolkit and explains how to use this toolkit.

## TARGET AUDIENCE


This user manual is intended for .NET developers of OPC UA Server applications. It assumes that you have a working knowledge of OPC UA and programming with the .NET languages.

## RELATED DOCUMENTATION

OPC Foundation ([www.opcfoundation.org](http://www.opcfoundation.org))

- OPC UA Specifications

## DOCUMENT CONVENTIONS

Convention	Description
Monospaced type	Indicates a file reference.
	Information to be noted.

## CUSTOMER SUPPORT SERVICES

Phone	Email
<b>Americas:</b> +1 713 609 9208  <b>Europe-Africa-Middle East</b> +216 71 195 360	Support: <a href="mailto:customerservice@integrationobjects.com">customerservice@integrationobjects.com</a>  Sales: <a href="mailto:sales@integrationobjects.com">sales@integrationobjects.com</a>  Online: <a href="http://www.integrationobjects.com">www.integrationobjects.com</a>

# INTRODUCTION

## 1. Overview

Integration Objects' OPC UA Server Toolkit is an API that handles all OPC UA details necessary to communicate with OPC UA clients. It is a tool for fast and easy programming of OPC UA server applications using the .NET framework.

Using this toolkit, developers can build their own OPC UA server applications easily using C# .NET or VB .NET and without having to be concerned with the details of the OPC UA standard. The generated .NET custom applications can be accessed locally or remotely from any OPC UA clients via its endpoint URLs.

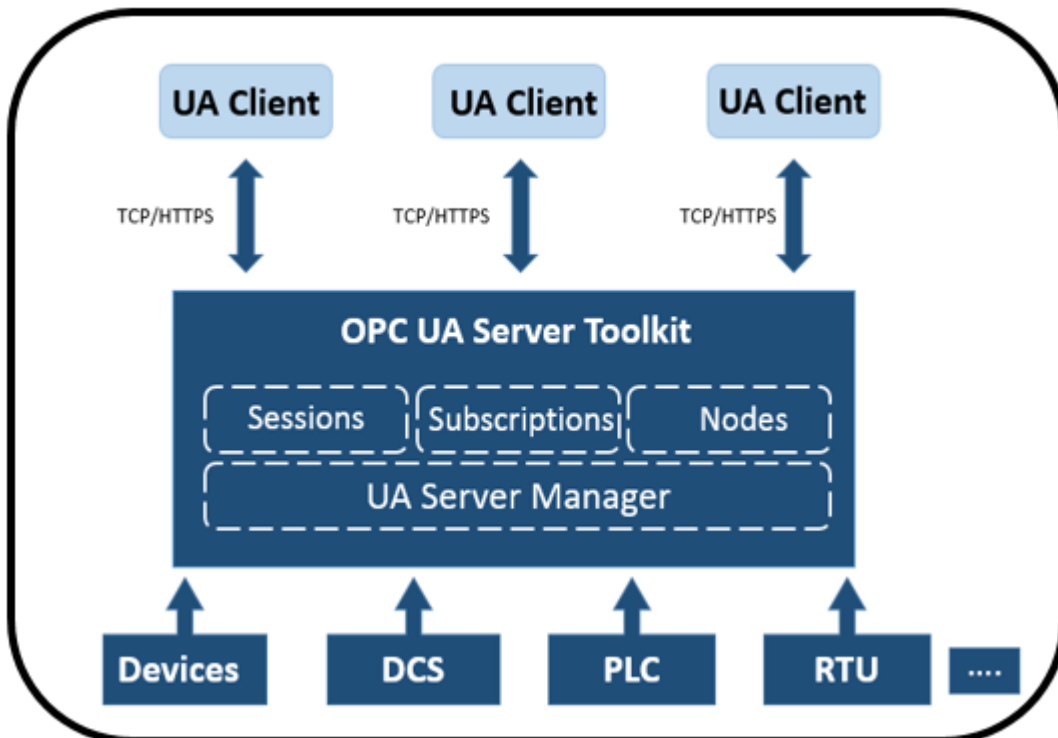


Figure 1: OPC UA Server Toolkit Overview

## 2. Features

The main features of OPC UA Server Toolkit are:

- Support of OPC UA specifications. This toolkit is compliant with OPC UA 1.03.
- Access to both real time and historical data.
- Support methods functionality.
- Accessible locally and via remote to multiple OPC UA clients.
- OPC UA server functionalities:
  - A secure session with OPC UA client.
  - Creating server address space.
  - Update real-time data values.
- Support of 32 and 64 bit applications.
- Royalty free runtime distribution.
- Support of Visual Studio 2017.

## 3. Operating Systems Compatibility

This Toolkit supports the following operating systems:

- Windows 10
- Windows 8
- Windows 7
- Windows Server 2019
- Windows Server 2016
- Windows Server 2012
- Windows Server 2008

## 4. OPC Compatibility

- OPC Unified Architecture 1.02
- OPC Unified Architecture 1.03

# GETTING STARTED

## 1. Pre-Installation Considerations

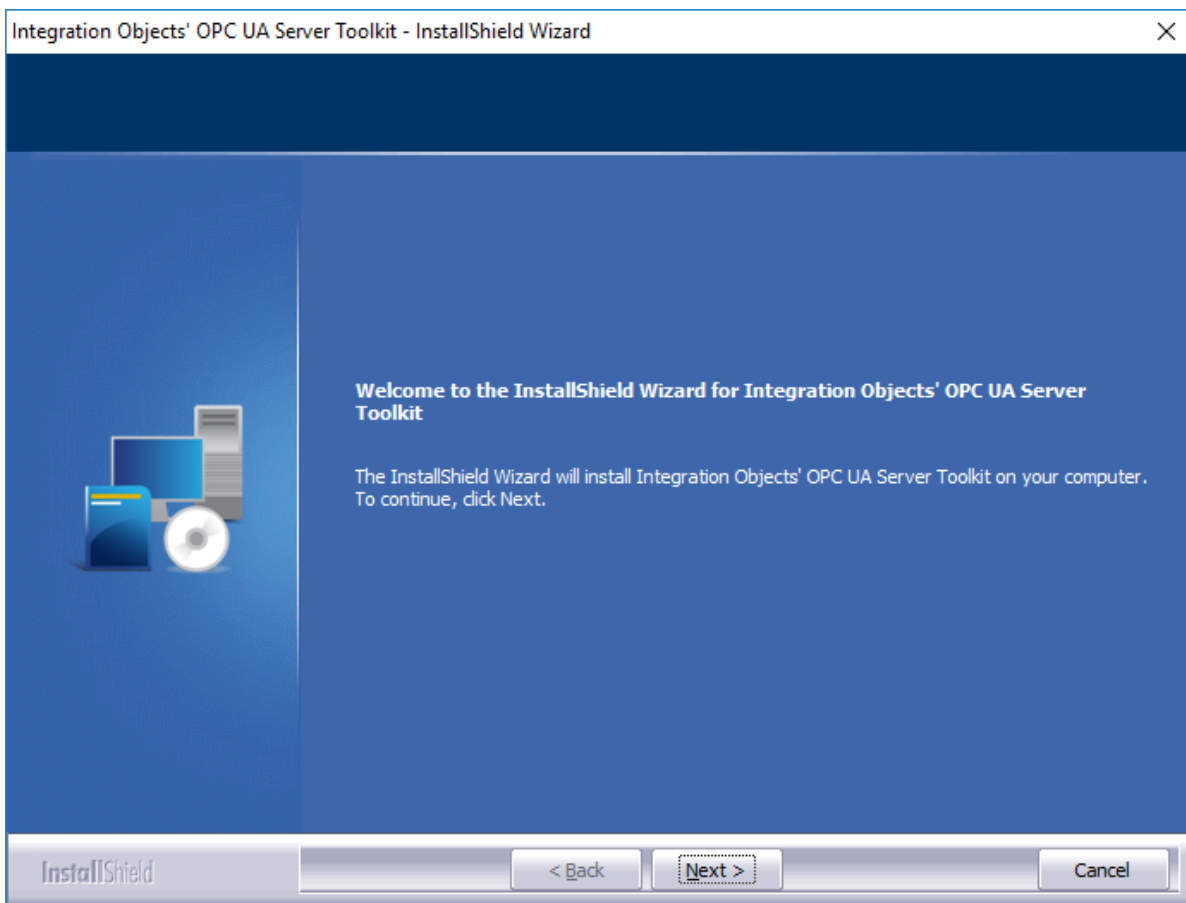
In order to properly use the OPC UA Server Toolkit or run an OPC UA Server developed using the OPC UA Server Toolkit, you need to install the following software components on the target system:

- .NET Framework version 4.6 or higher.

## 2. Installation

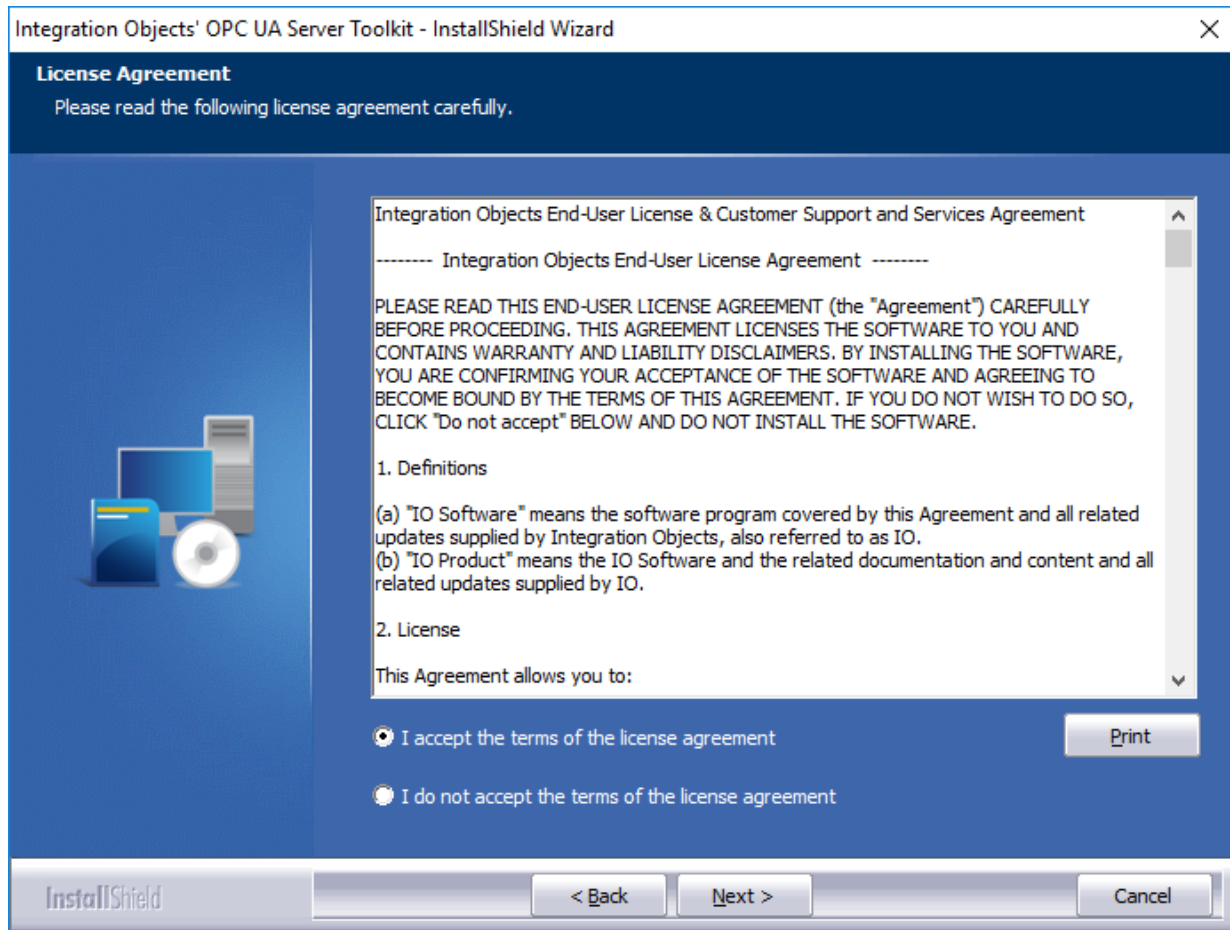
To install the OPC UA Server toolkit:

1. Right click the downloaded installation program and select “Run as administrator” from the displayed menu. The installation wizard will then guide you through the different installation steps.



**Figure 2: Installation Welcome Dialog**

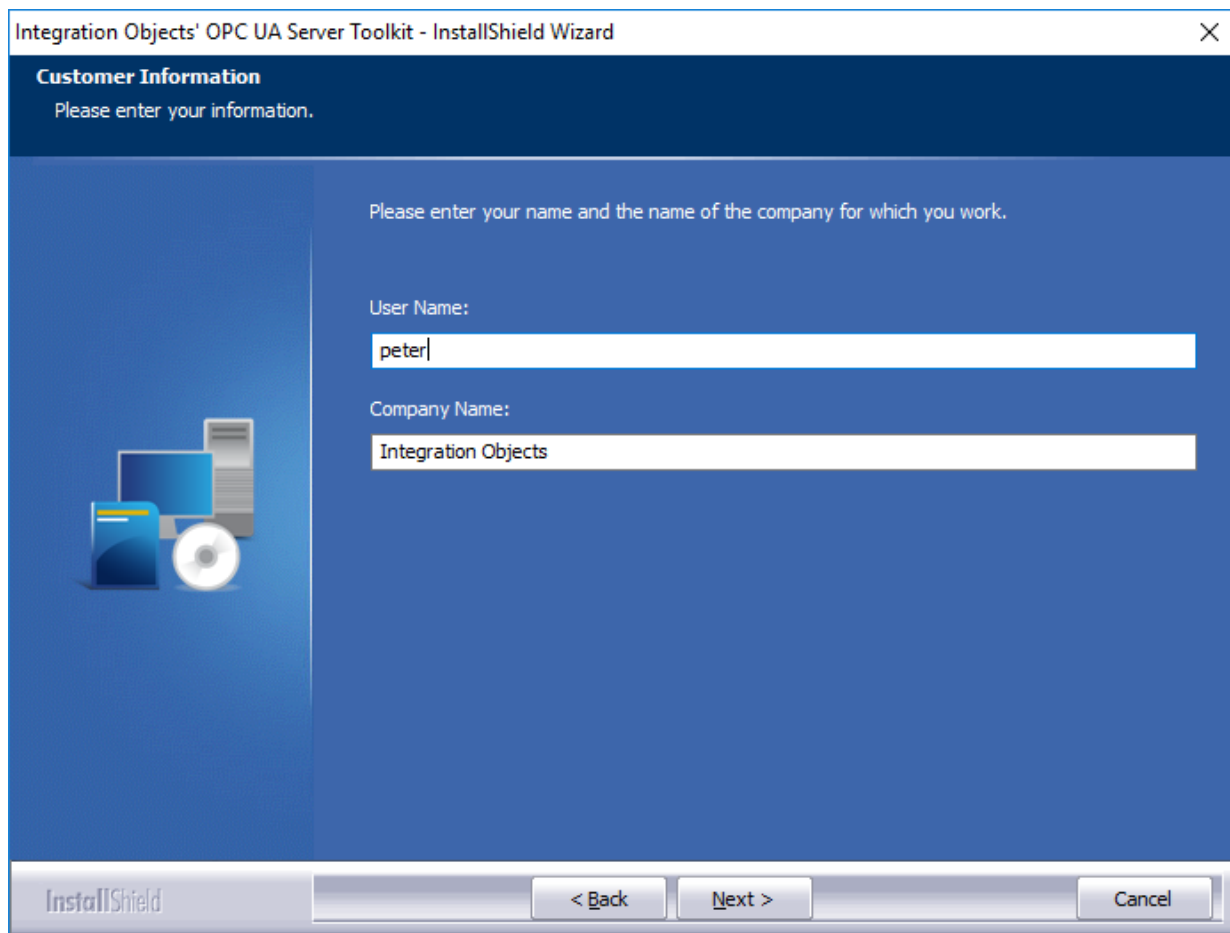
2. Click the **Next** button. The license agreement will be displayed:



**Figure 3: 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 will appear:





Integration Objects' OPC UA Server Toolkit - InstallShield Wizard

**Customer Information**  
Please enter your information.

Please enter your name and the name of the company for which you work.

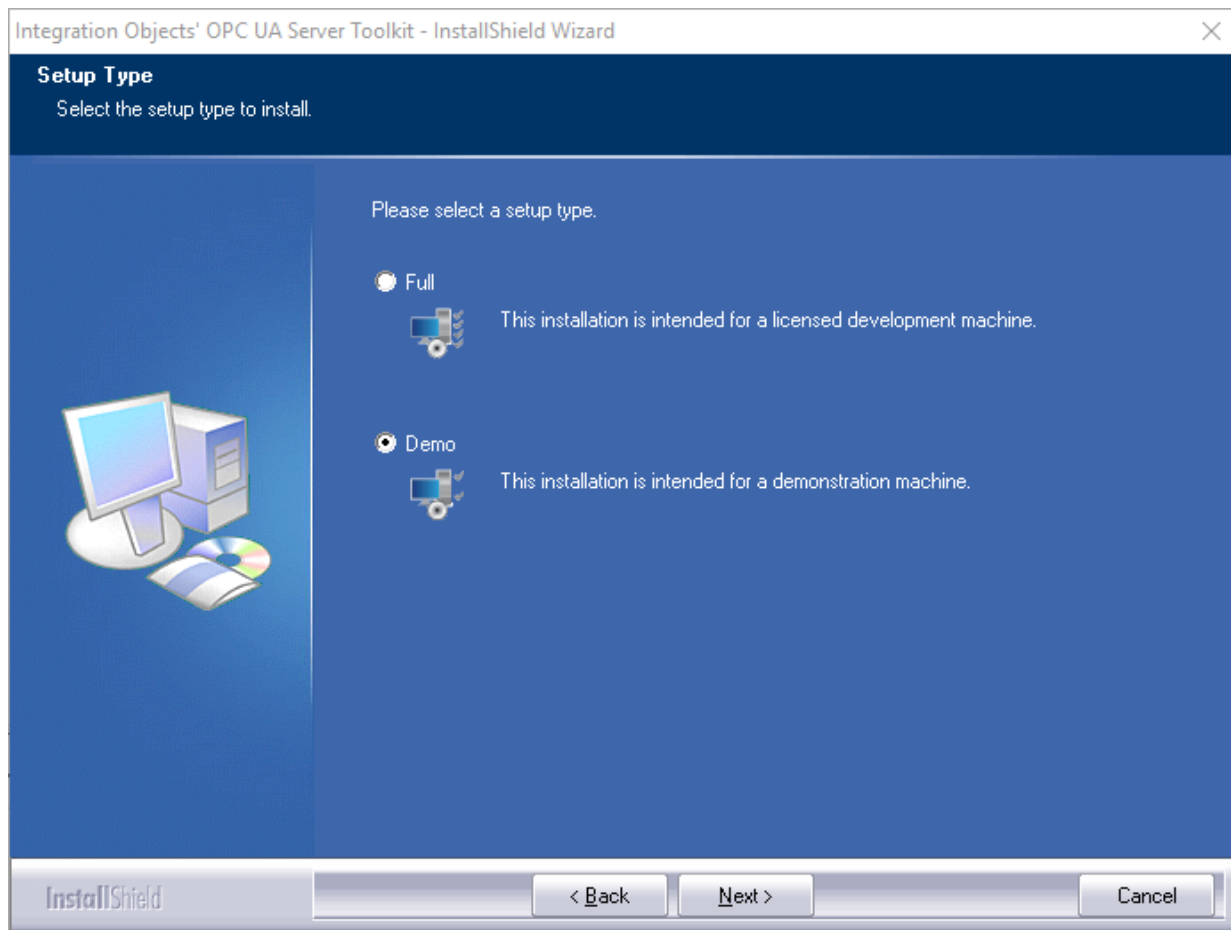
User Name:  
peter

Company Name:  
Integration Objects

InstallShield < Back Next > Cancel

**Figure 4: Customer Information Dialog**

4. Add the user name and the company name and then click the **Next** button. The dialog box for selecting the setup type will be displayed:

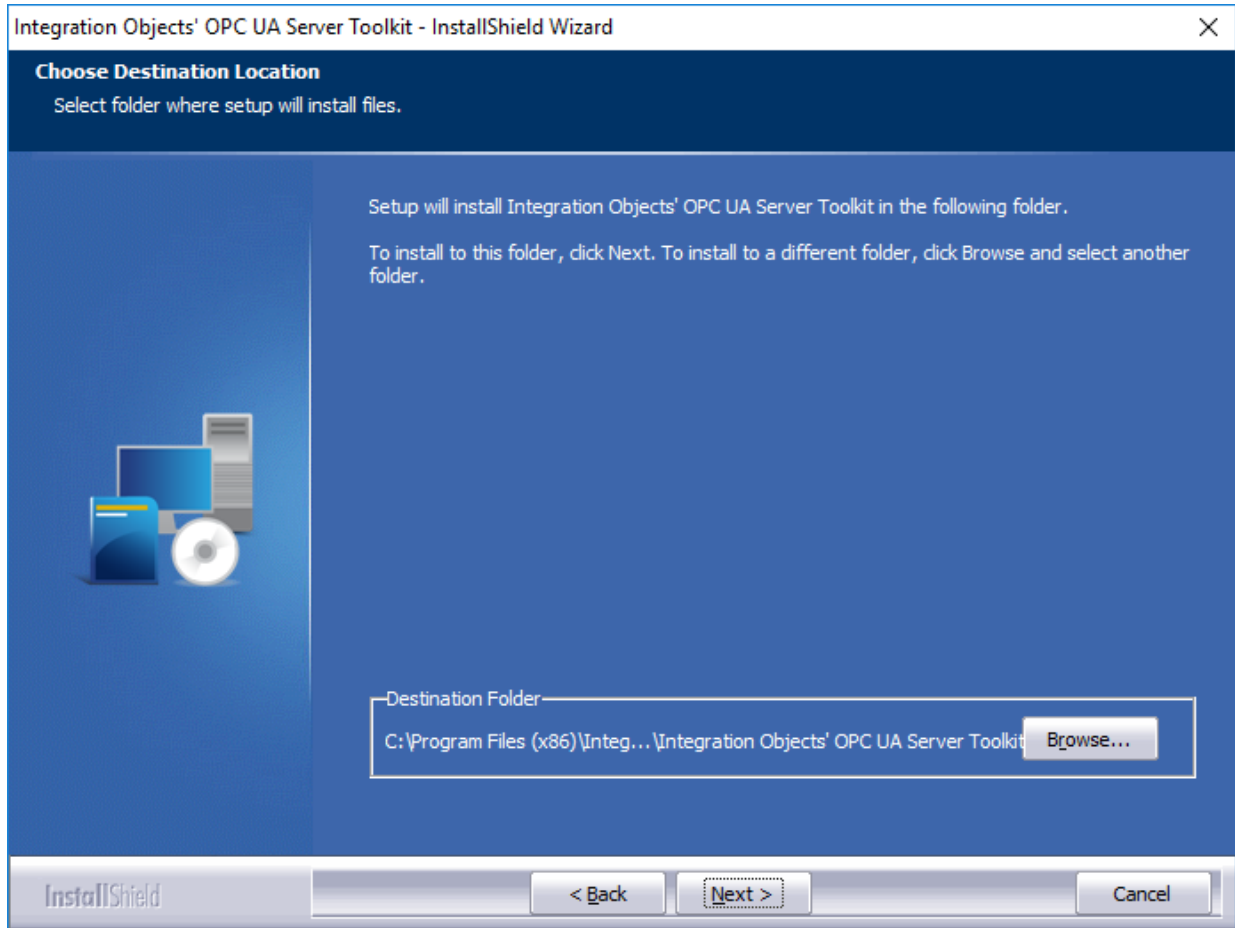


**Figure 5: Setup Type Dialog**

5. Make sure to select:

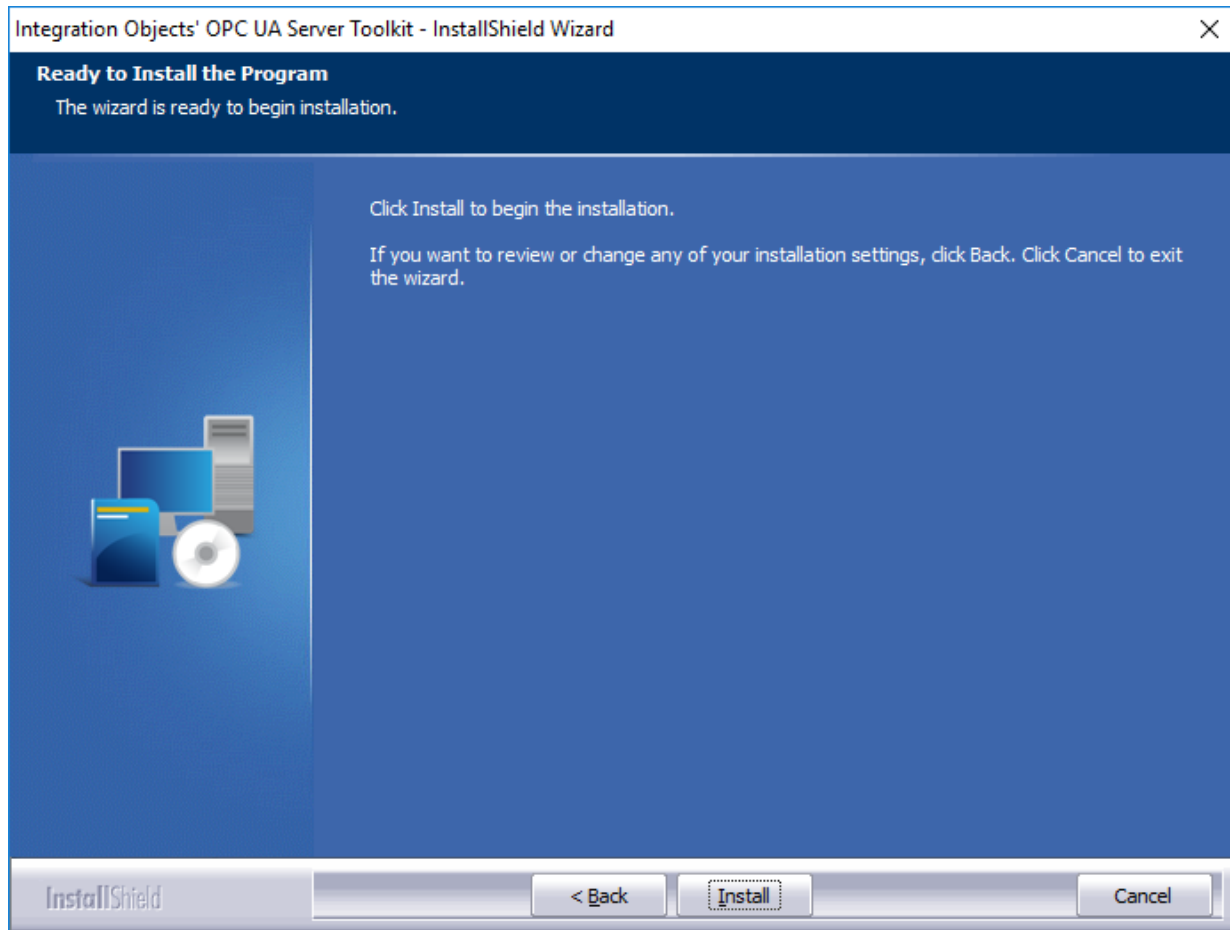
- The Demo version option if you are evaluating the OPC UA Server Toolkit. This will allow you to use an evaluation license of the toolkit that is valid for 30 days and limited to 2 hour the runtime.
- The Full version option for a licensed development machine installation.

After selecting the setup type, click the **Next** button. The dialog box for choosing the destination folder will be displayed:



**Figure 6: Choose Destination Location Dialog**

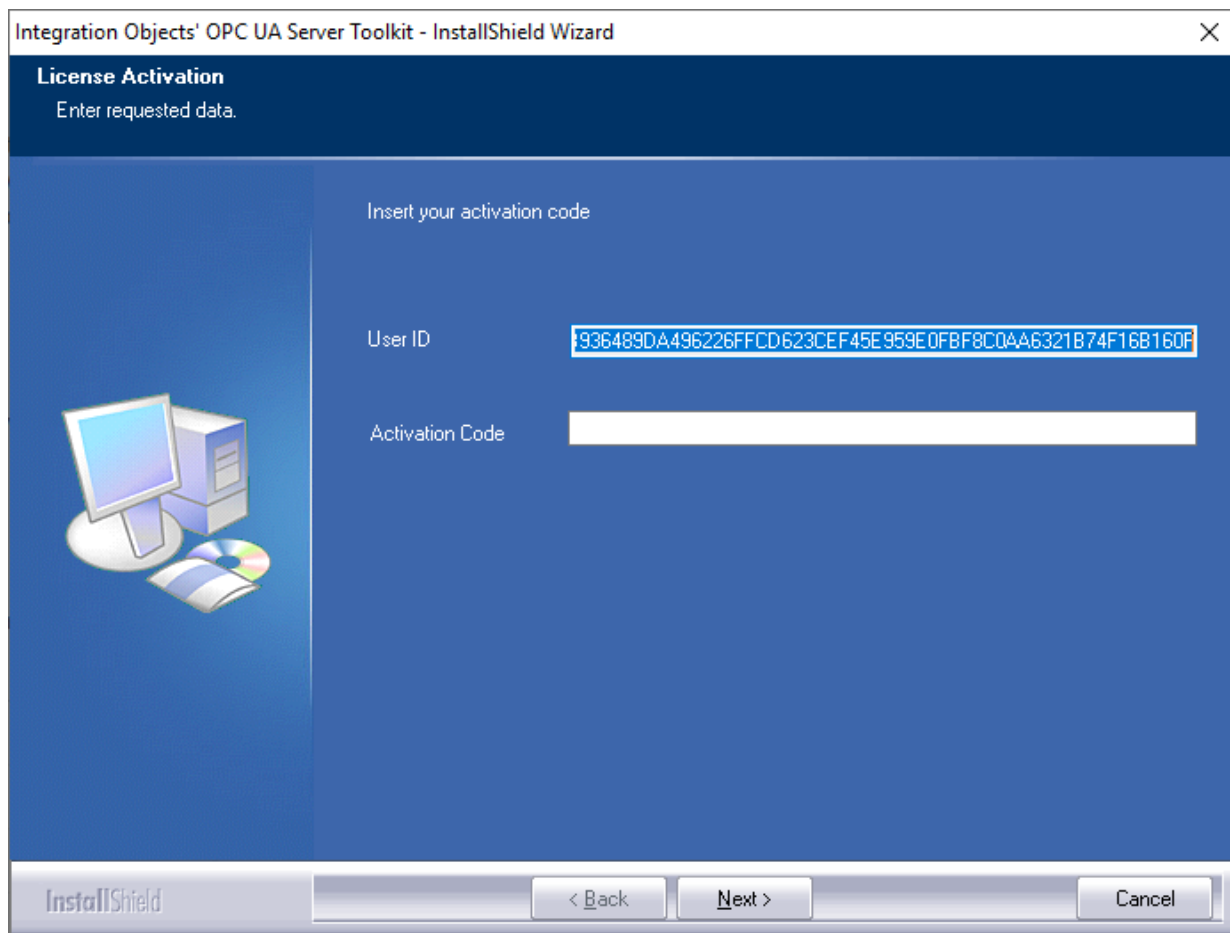
6. Click the **Next** button to use the default destination folder and continue the installation, or you can click the **Browse** button to select a different destination folder. The installation dialog box will then appear:



**Figure 7: Ready to Install the Program Dialog**

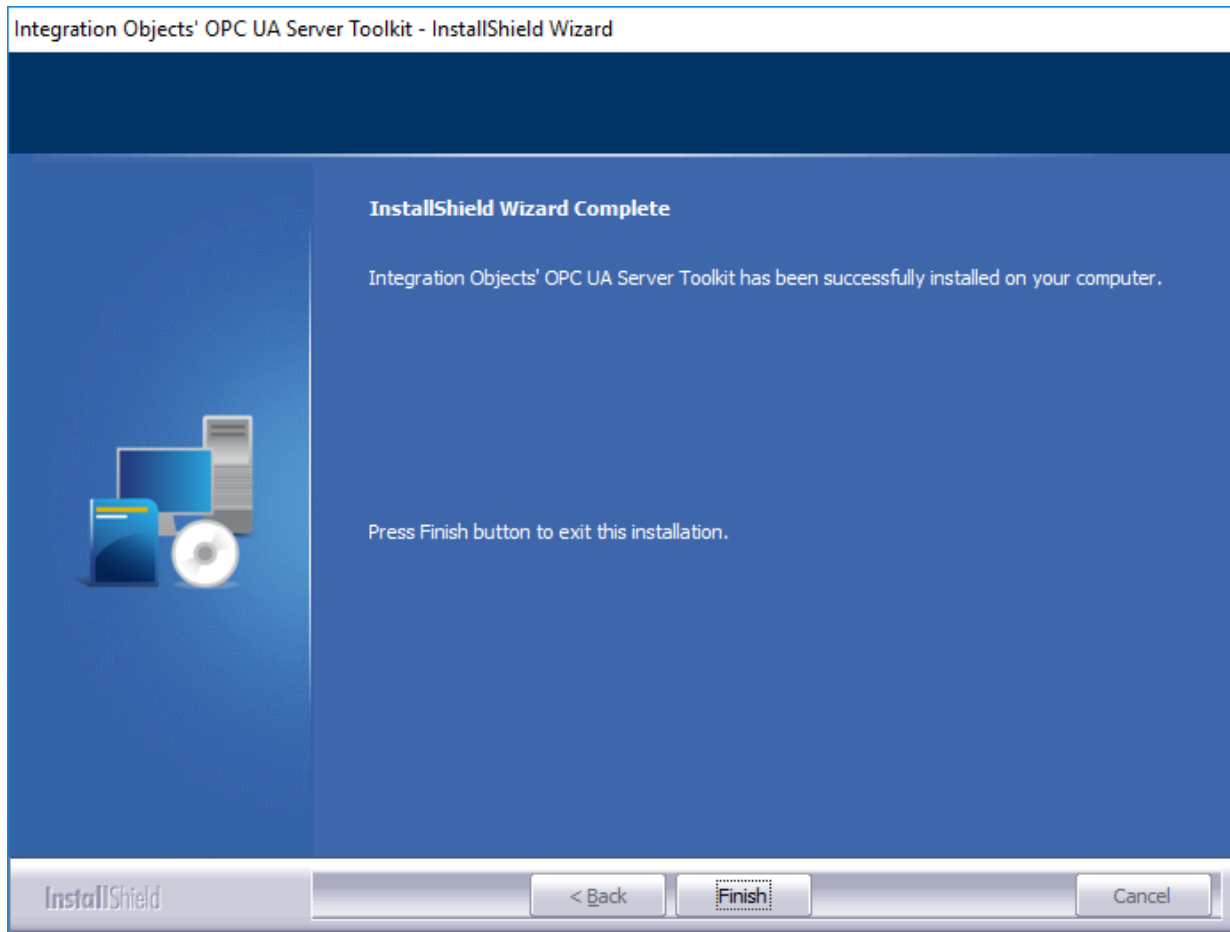
7. Click the **Install** button to start installation.

If the **Full** option was selected as setup type, then the next step is to enter the received code in the **Activation Code** field and click the **Next** button. Copy the “User ID” and send it to Integration Objects’ sales team ([sales@integrationobjects.com](mailto:sales@integrationobjects.com)) in a text format and they will get back to you with the “Activation Code”.



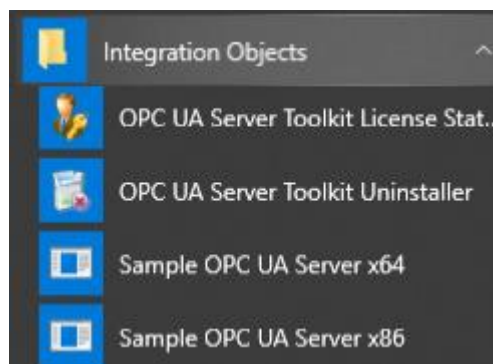
**Figure 8 : License Activation Dialog**

8. Click the **Finish** button. Now, the OPC UA Server Toolkit is properly installed in your development machine.



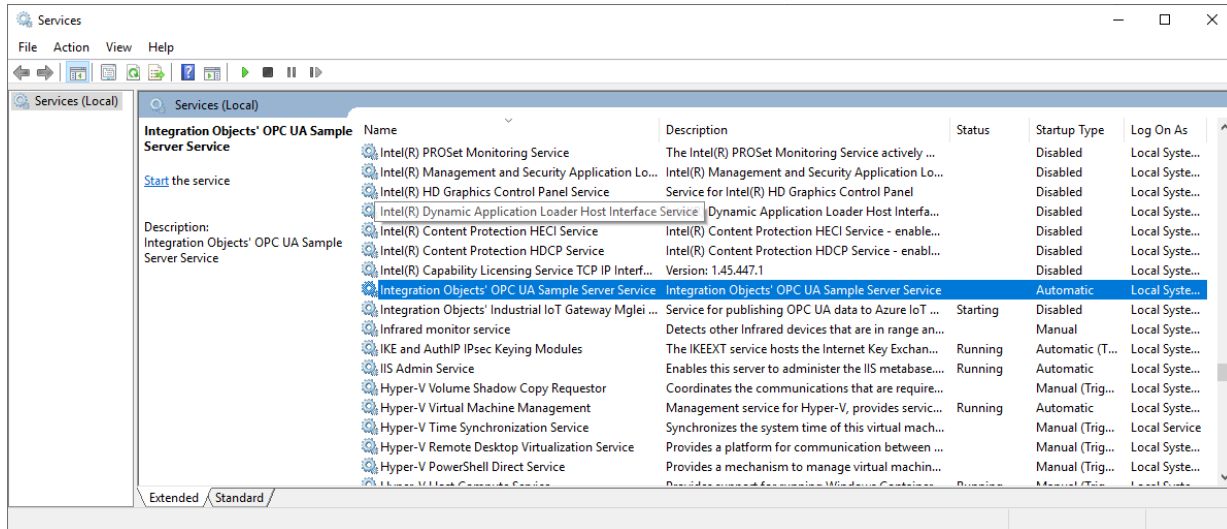
**Figure 9: Installation Completed Dialog**

Once the installation is complete, you will have the following shortcuts in your start menu:



**Figure 10: OPC UA Server Toolkit Start Menu**

And you will have an “OPC UA Sample Server service” available on the Windows services:



**Figure 11: OPC UA Sample Server Service**

You will also get the following files in your system under the selected destination folder:

Files	Description
DLL Files	<ul style="list-style-type: none"> <li>• Config.ini.</li> <li>• IntegrationObjects.Logger.SDK.dll.</li> <li>• IntegrationObjects.Opc.Ua.Server.Toolkit.dll.</li> <li>• License.dll.</li> <li>• libuv.dll</li> <li>• IntegrationObjects.Opc.Ua.Utilities.dll.</li> <li>• IntegrationObjects.Opc.Ua.Core.dll.</li> <li>• IntegrationObjects.Opc.Ua.Configuration.dll</li> <li>• IntegrationObjects.Opc.Ua.Server.dll</li> <li>• The UA XML configuration file (XXXX.Config.xml, where XXXX is the name of your OPC UA server application).</li> <li>• BouncyCastle.Crypto.dll.</li> <li>• Microsoft.AspNetCore.Hosting.Abstractions.dll.</li> <li>• Microsoft.AspNetCore.Hosting.dll.</li> <li>• Microsoft.AspNetCore.Hosting.Server.Abstractions.dll.</li> <li>• Microsoft.AspNetCore.Http.Abstractions.dll.</li> <li>• Microsoft.AspNetCore.Http.dll.</li> <li>• Microsoft.AspNetCore.Http.Features.dll.</li> <li>• Microsoft.AspNetCore.Server.Kestrel.dll.</li> <li>• Microsoft.AspNetCore.Server.Kestrel.Https.dll.</li> <li>• Microsoft.AspNetCore.WebUtilities.dll.</li> <li>• Microsoft.Extensions.Configuration.Abstractions.dll.</li> </ul>

	<ul style="list-style-type: none"> <li>• Microsoft.Extensions.Configuration.dll.</li> <li>• Microsoft.Extensions.Configuration.EnvironmentVariables.dll.</li> <li>• Microsoft.Extensions.DependencyInjection.Abstractions.dll.</li> <li>• Microsoft.Extensions.DependencyInjection.dll.</li> <li>• Microsoft.Extensions.FileProviders.Abstractions.dll.</li> <li>• Microsoft.Extensions.FileProviders.Physical.dll.</li> <li>• Microsoft.Extensions.FileSystemGlobbing.dll.</li> <li>• Microsoft.Extensions.Logging.Abstractions.dll.</li> <li>• Microsoft.Extensions.Logging.dll.</li> <li>• Microsoft.Extensions.ObjectPool.dll.</li> <li>• Microsoft.Extensions.Options.dll.</li> <li>• Microsoft.Extensions.PlatformAbstractions.dll.</li> <li>• Microsoft.Extensions.Primitives.dll.</li> <li>• Microsoft.Net.Http.Headers.dll.</li> <li>• Newtonsoft.Json.dll.</li> <li>• System.Buffers.dll.</li> <li>• System.Collections.Immutable.dll.</li> <li>• System.Diagnostics.DiagnosticSource.dll.</li> <li>• System.Numerics.Vectors.dll.</li> <li>• System.Reflection.Metadata.dll.</li> <li>• System.Runtime.CompilerServices.Unsafe.dll.</li> <li>• System.Runtime.InteropServices.RuntimeInformation.dll.</li> <li>• System.Text.Encodings.Web.dll.</li> <li>• System.Threading.Tasks.Extensions.dll.</li> </ul>
OPC Sample Demos	Contains the OPC UA Server samples for both C# .NET and VB .NET programming language in x86 and x64 architectures
OPC Sample Projects	Contains the Visual Studio 2017 projects of the OPC UA Server samples.
Other files	<ul style="list-style-type: none"> <li>• OPC UA Server Toolkit User Guide (this guide)</li> <li>• OPC UA Server Toolkit Quick User Guide</li> <li>• License authorization tool: Indicates the license status of each installed feature.</li> <li>• OPC UA Server Toolkit Uninstaller: used to uninstall the OPC UA Server Toolkit</li> </ul>
Components	Contains the OPC UA Local Discovery Server installation program

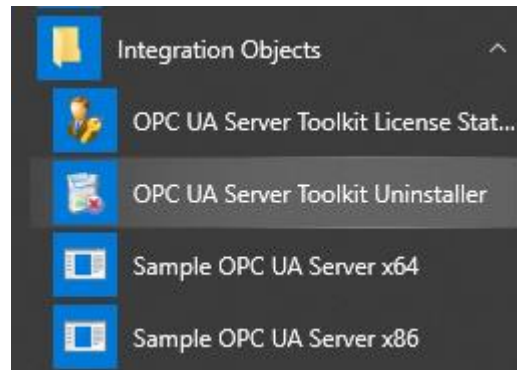
**Table 1: Files Included in the Distribution**

### 3. Uninstallation

To uninstall the OPC UA Server Toolkit, you have two options:

1. From the **Start** menu, select the “OPC UA Server Toolkit uninstaller”.





**Figure 12: Uninstaller Shortcut in the Start Menu**

The uninstaller wizard will be automatically launched and will take you through the different steps of the software removal.

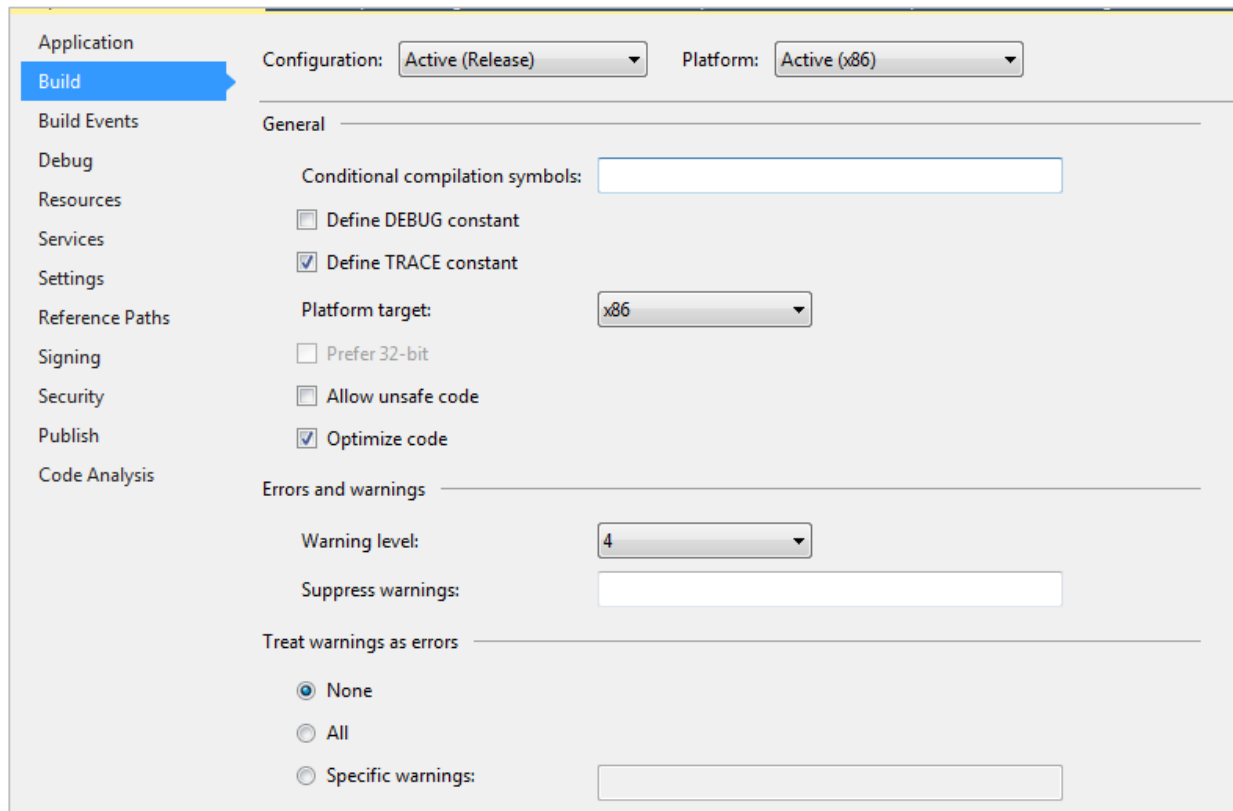
2. From Add/Remove Programs:
  - a. Go to the start menu.
  - b. Type “Control Panel”.
  - c. Select “Programs and Features”
  - d. Choose Integration objects’ OPC UA Server Toolkit from the programs’ list.
  - e. Click on the **Uninstall** button.

## 4. Compiling and Linking Applications

This section details the steps to follow in order to compile and correctly link applications to develop your OPC UA server application using Integration Objects’ OPC UA Server Toolkit and Microsoft Visual Studio 2017.

- **Step 1: Select your Platform**

For users who have to build the application in a **32-bit** machine, the target platform has to be **x86** as illustrated in the screenshot below:



**Figure 13: Platform Target for 32-bit Machine**

For users who have to build the application in a **64-bit** machine, the target platform has to be **x64** as illustrated in the screenshot below:

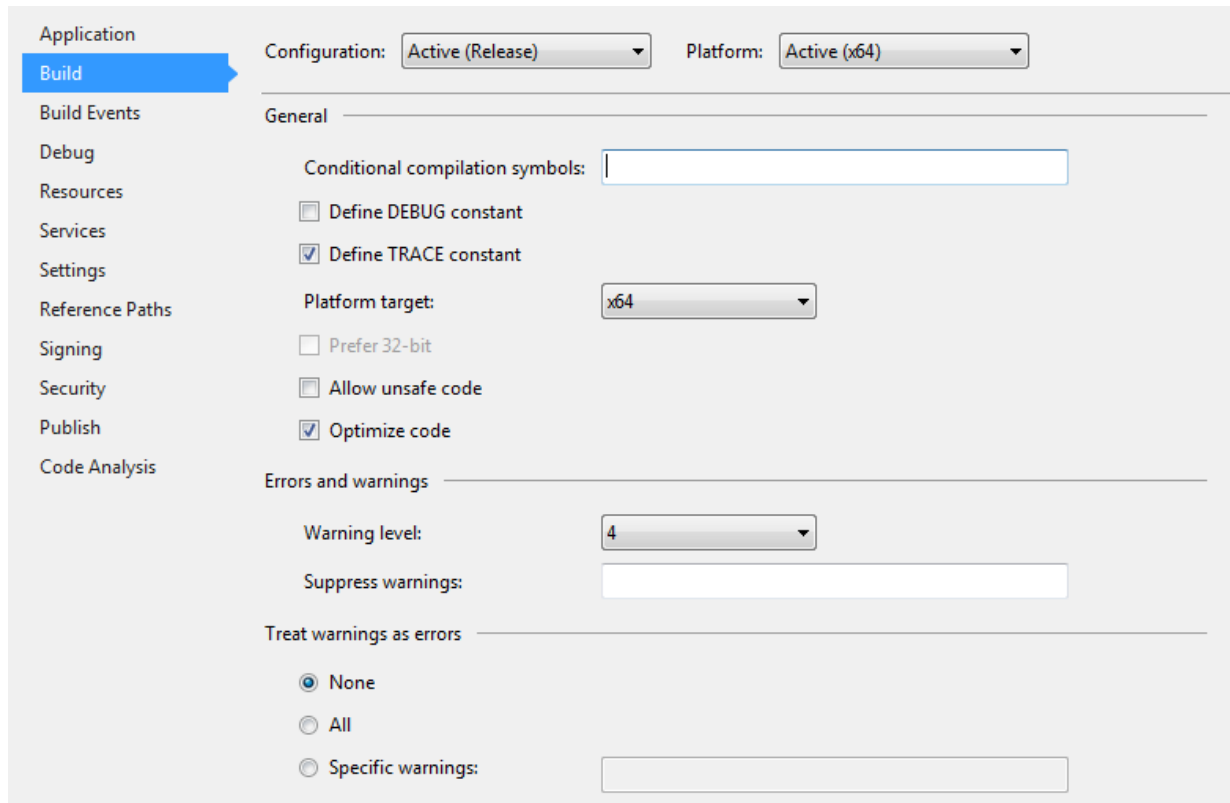
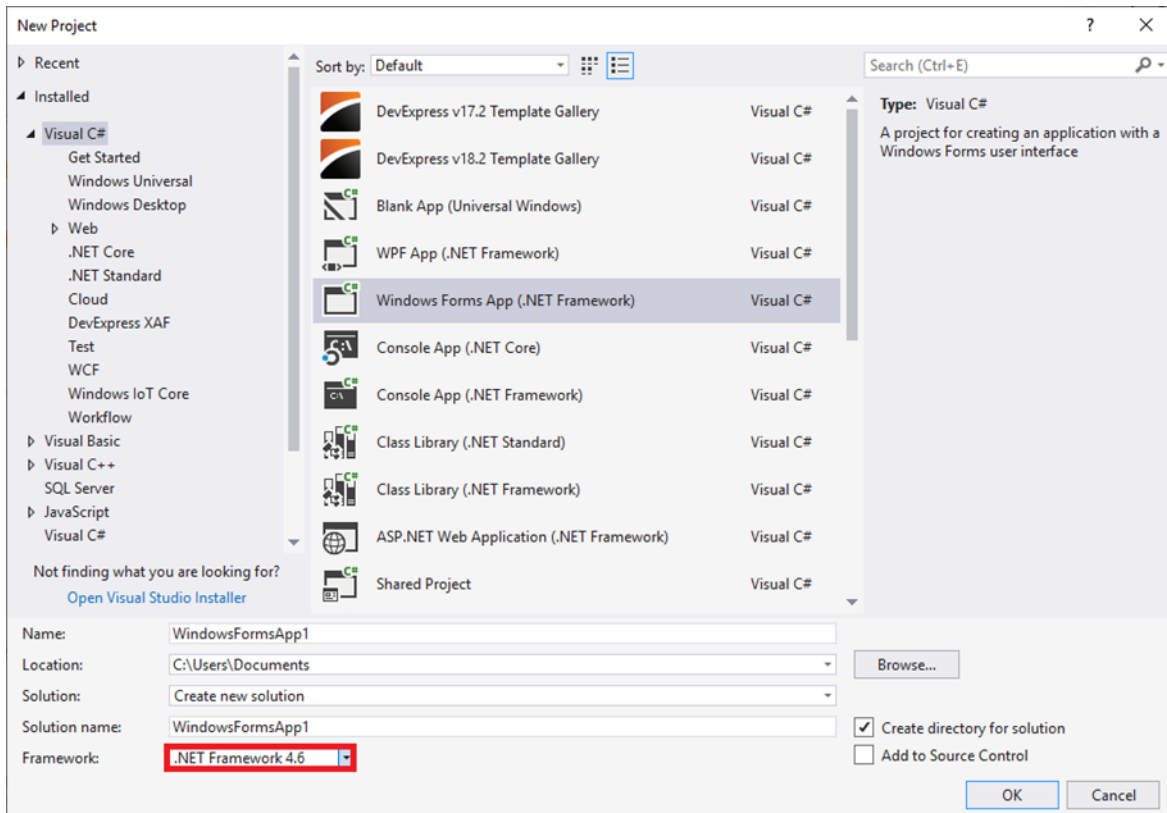


Figure 14: Platform Target for 64-bit Machine

- **Step 2: Create your Project**

Start Visual Studio 2017 and choose **New Project**. The following window will be displayed:



**Figure 15: New Windows Form Project**

Choose Visual C# **Windows Forms Application** Project and then click **OK**.

A project named WindowFormsApplication with a form called Form1 will be automatically created.

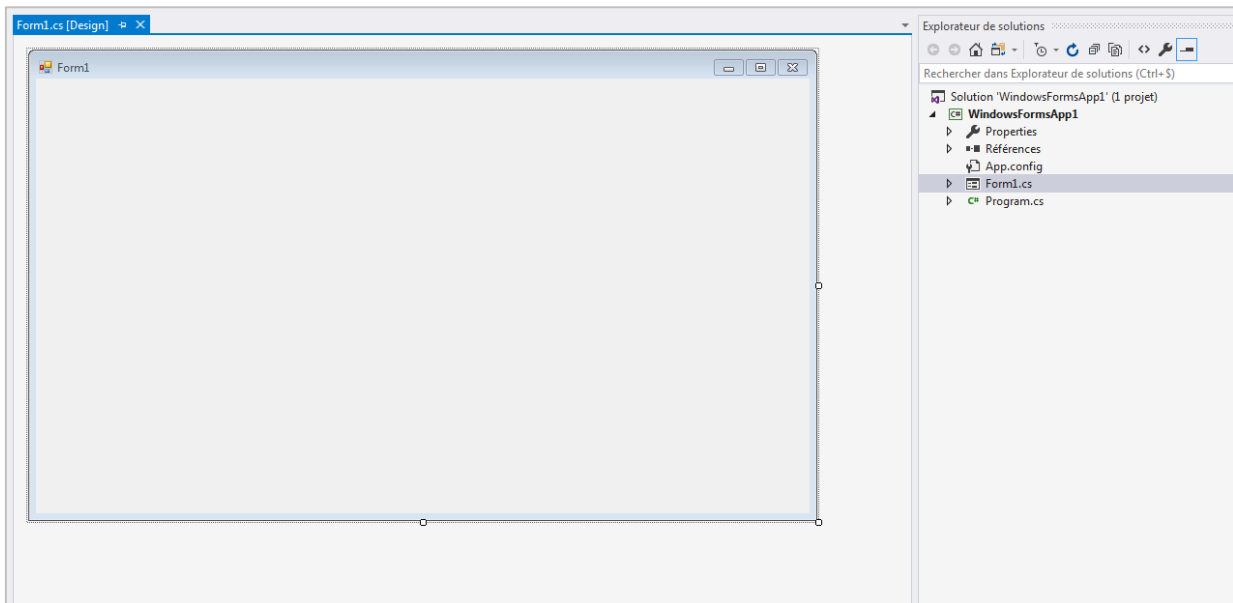


Figure 16: Windows Form Project

- **Step 3: Add your References**

1. Add reference to the OPC UA Server Toolkit as shown below.

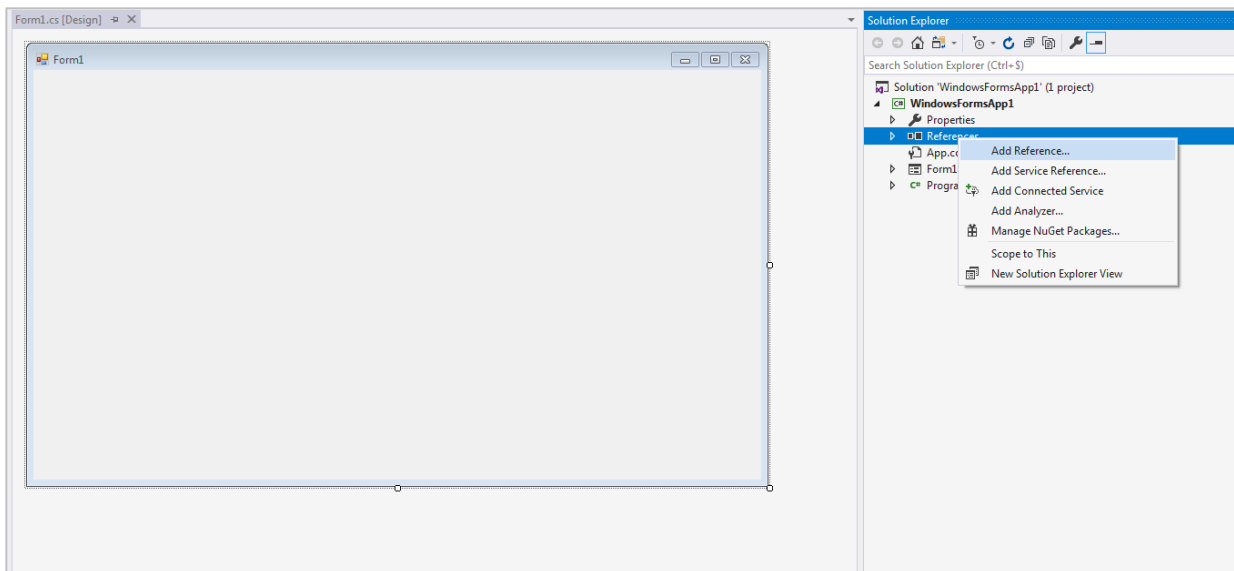


Figure 17: Add Reference

2. Select Browse tab from the displayed Add Reference window.
3. Select **IntegrationObjects.Opc.Ua.Server.Toolkit.dll** located under the **bin** folder in the installation folder.

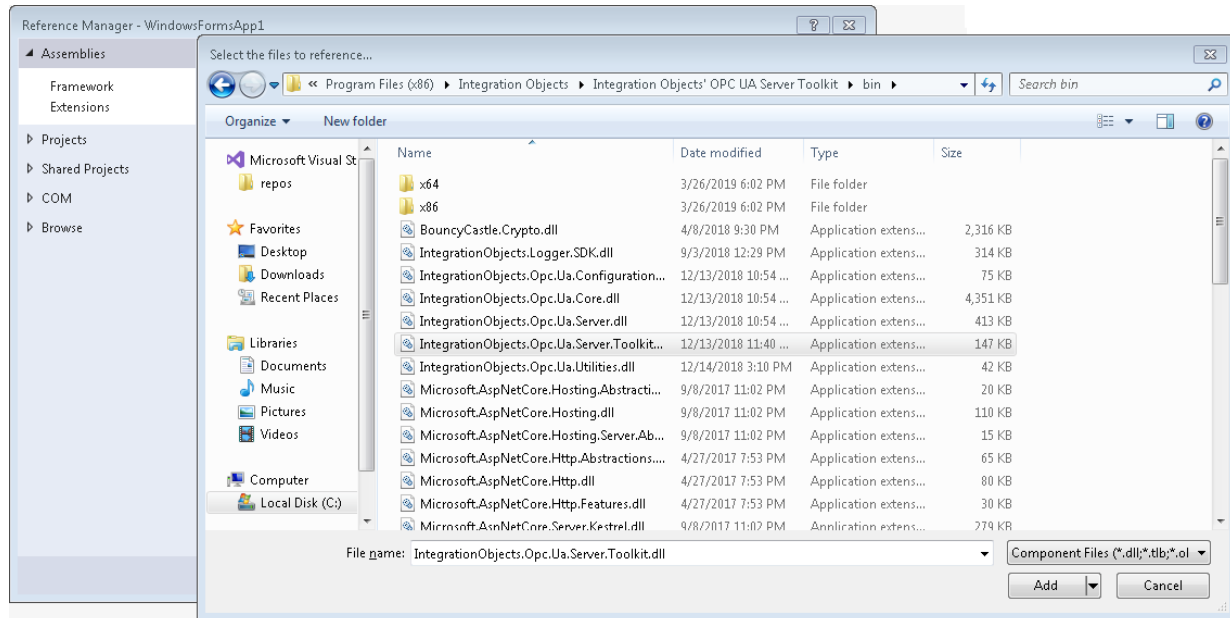


Figure 18: Choosing a Reference

## 5. Runtime Deployment Procedure

In order to deploy the developed server application from the development machine to the runtime machine, follow the steps below:

1. Create a new folder.
2. Copy the following files:
  - The application executable and any other custom assembly dependencies.
  - The UA XML configuration file (XXXX.Config.xml, where XXXX is the name of your OPC UA server application).
  - Config.ini.
  - IntegrationObjects.Logger.SDK.dll.
  - IntegrationObjects.Opc.Ua.Server.Toolkit.dll.
  - IntegrationObjects.Opc.Ua.Utilities.dll.
  - IntegrationObjects.Opc.Ua.Core.dll.
  - IntegrationObjects.Opc.Ua.Configuration.dll
  - IntegrationObjects.Opc.Ua.Server.dll
  - License.dll.
  - libuv.dll
  - BouncyCastle.Crypto.dll.
  - Microsoft.AspNetCore.Hosting.Abstractions.dll.

- Microsoft.AspNetCore.Hosting.dll.
- Microsoft.AspNetCore.Hosting.Server.Abstractions.dll.
- Microsoft.AspNetCore.Http.Abstractions.dll.
- Microsoft.AspNetCore.Http.dll.
- Microsoft.AspNetCore.Http.Features.dll.
- Microsoft.AspNetCore.Server.Kestrel.dll.
- Microsoft.AspNetCore.Server.Kestrel.Https.dll.
- Microsoft.AspNetCore.WebUtilities.dll.
- Microsoft.Extensions.Configuration.Abstractions.dll.
- Microsoft.Extensions.Configuration.dll.
- Microsoft.Extensions.Configuration.EnvironmentVariables.dll.
- Microsoft.Extensions.DependencyInjection.Abstractions.dll.
- Microsoft.Extensions.DependencyInjection.dll.
- Microsoft.Extensions.FileProviders.Abstractions.dll.
- Microsoft.Extensions.FileProviders.Physical.dll.
- Microsoft.Extensions.FileSystemGlobbing.dll.
- Microsoft.Extensions.Logging.Abstractions.dll.
- Microsoft.Extensions.Logging.dll.
- Microsoft.Extensions.ObjectPool.dll.
- Microsoft.Extensions.Options.dll.
- Microsoft.Extensions.PlatformAbstractions.dll.
- Microsoft.Extensions.Primitives.dll.
- Microsoft.Net.Http.Headers.dll.
- Newtonsoft.Json.dll.
- System.Buffers.dll.
- System.Collections.Immutable.dll.
- System.Diagnostics.DiagnosticSource.dll.
- System.Numerics.Vectors.dll.
- System.Reflection.Metadata.dll.
- System.Runtime.CompilerServices.Unsafe.dll.
- System.Runtime.InteropServices.RuntimeInformation.dll.
- System.Text.Encodings.Web.dll.
- System.Threading.Tasks.Extensions.dll.

3. Copy the folder to the runtime machine.



**Make sure that the OPC UA Server Toolkit is not installed in the runtime machine and that the path of the application folder does not include the key words “Debug” or “Release”.**

# USING THE OPC UA SERVER TOOLKIT

## 1. Initialization of the API

After the DLL initialization, the server application holds responsibility for properly initializing the API. To do so, the **UAServerManager** class contains the basic functions needed for the initialization.

## 2. UAServerManager Class

### 2.1. Constructor

The server application is responsible for properly initializing the OPC UA Server Toolkit using the **UAServerManager** class as follows:

```
UAServerManager objUAManager = new UAServerManager ();
```

### 2.2. Functions

#### 2.2.1. Address Space Management

##### 2.2.1.1. SetDARootFolder

- **Description**

This function sets the Real Time Root Folder.

```
void SetDARootFolder(Folder rootFolder)
```

- **Parameters**

The following table describes the parameters of the SetDARootFolder function.

In/Out	Parameter	Description
In	rootFolder	The folder object that contains all variables and folders related to the real time data.

Table 2 : Parameters of SetDARootFolder



### 2.2.1.2. SetHDARootFolder

- **Description**

This function sets the Historical Data Root Folder.

```
void SetHDARootFolder(Folder rootFolder)
```

- **Parameters**

The following table describes the parameters of the SetHDARootFolder function.

In/Out	Parameter	Description
In	RootFolder	The folder object that contains all variables and folders of related to the historical data.

**Table 3: Parameters of SetHDARootFolder**

### 2.2.1.3. SetMethodsRootFolder

- **Description**

This function sets the OPC UA Methods Root Folder.

```
void SetMethodsRootFolder(Folder rootFolder)
```

- **Parameters**

The following table describes the parameters of the SetMethodsRootFolder function.

In/Out	Parameter	Description
In	rootFolder	The folder object that contains all methods and folders of OPC UA Methods.

**Table 4: Parameters of SetMethodsRootFolder**

### 2.2.1.4. AddVariableToFolder

- **Description**

This function adds a variable to a folder.

```
void AddVariableToFolder(Folder folder, Variable variable)
```

- **Parameters**

The following table describes the parameters of the AddVariableToFolder function.

In/Out	Parameter	Description
--------	-----------	-------------

In	Folder	The destination folder.
In	Variable	The variable to be added.

**Table 5: Parameters of AddVariableToFolder**

### 2.2.1.5. AddVariablesToFolder

- **Description**

This function adds a set of variables to a folder.

```
void AddVariablesToFolder(Folder folder, List<Variable> variables)
```

- **Parameters**

The following table describes the parameters of the AddVariablesToFolder function.

In/Out	Parameter	Description
In	Folder	The destination folder.
In	Variables	The list of variables to be added.

**Table 6: Parameters of AddVariablesToFolder**

### 2.2.1.6. AddMethodsToFolder

- **Description**

This function adds a list of methods to a folder.

```
void AddMethodsToFolder(Folder folder, List<Method> Methods)
```

- **Parameters**

The following table describes the parameters of the AddMethodsToFolder function.

In/Out	Parameter	Description
In	Folder	The destination folder.
In	Methods	The list of methods to be added.

**Table 7: Parameters of AddMethodsToFolder**

### 2.2.1.7. AddFolderToFolder

- **Description**

This function adds a folder to another folder.

```
void AddFolderToFolder(Folder folder, Folder FolderToAdd)
```

- **Parameters**

The following table describes the parameters of the AddFolderToFolder function.

In/Out	Parameter	Description
In	Folder	The destination folder.
In	foldertoAdd	The added list of methods.

**Table 8: Parameters of AddFolderToFolder**

### 2.2.1.8. UpdateDAVariable

- **Description**

This function updates a real time data variable value.

`bool UpdateDAVariable(string TagName, object TagValue, DateTime Timestamp, StatusCode statusCode)`

- **Parameters**

The following table describes the parameters of the UpdateDAVariable function.

In/Out	Parameter	Description
In	TagName	The Tag Name to be updated.
In	TagValue	The Tag Value to be updated.
In	Timestamp	The time stamp to be updated.
In	statusCode	The status code to be updated.

**Table 9: Parameters of UpdateDAVariable**

- **Returned results**

Return Code	Description
True	The operation was successful.
False	The operation failed.

**Table 10: Returned Results of UpdateDAVariable**

## 2.2.2. Start/Stop Server

### 2.2.2.1. StartServer

- **Description**

This function starts the OPC UA server.

```
bool StartServer(out string strError)
```

- **Parameters**

The following table describes the parameters of the StartServer function.

In/Out	Parameter	Description
Out	strError	If the operation fails, this function will return an error message. Otherwise, it will return an empty string.

**Table 11: Parameters of StartServer**

- **Returned results**

Return Code	Description
True	The operation was successful.
False	The operation failed.

**Table 12: Returned Results of StartServer**

### 2.2.2.2. StopServer

- **Description**

This function stops the OPC UA server.

```
bool StopServer(out string strError)
```

- **Parameters**

The following table describes the parameters of the StopServer function.

In/Out	Parameter	Description
Out	strError	If the operation fails, this function will return an error message. Otherwise, it will return an empty string.

**Table 13: Parameters of StopServer**

- **Returned results**

Return Code	Description
-------------	-------------

True	The operation was successful.
False	The operation failed.

**Table 14: Returned Results of StopServer**

## 2.2.3. Server Statistics Management

### 2.2.3.1. UpdateSessions

- **Description**

This function updates the current sessions.

`IList<Session> UpdateSessions()`

- **Returned results**

The following table describes the returned objects of the UpdateSessions function.

Type	Description
List<Session>	List of created sessions.

**Table 15: Returned Results of UpdateSessions**

### 2.2.3.2. UpdateSubscriptions

- **Description**

This function updates the current subscriptions.

`IList<Subscription> UpdateSubscriptions()`

- **Returned results**

The following table describes the returned objects of the UpdateSubscriptions function.

Type	Description
List<Subscription>	List of created subscription.

**Table 16: Returned Results of UpdateSubscriptions**

### 2.2.3.3. GetServerStatus

- **Description**

This function returns a string that describe the status of the Server.

`string GetServerStatus()`

- **Returned results**

The following table describes the parameters deployed and needed by the GetServerStatus function.

Return Code	Description
Running	The server is running.

Failed	The server failed.
NoConfiguration	The server is not configured.
Suspended	The server is suspended.
Shutdown	The server is shutdown.
Test	The server is in test mode. This means that the variables are disconnected from the source device but the OPC UA Server is operational.
CommunicationFault	Failed to communicate with the server.
Unknown	The server status is unknown

**Table 17: Returned Results of GetServerStatus**

#### 2.2.3.4. UpdateEndpoints

- **Description**

This function updates the available endpoints of the server via which a connection can be established.

`EndpointDescriptionCollection UpdateEndpoints()`

- **Returned results**

The following table describes the objects returned by the UpdateEndpoints function.

Type	Description
EndpointDescriptionCollection	A list of EndpointDescription containing descriptions about the endpoints used by the client to connect to the server

**Table 18: Returned Results of UpdateEndpoints**

## 2.2.4. Server Configuration

### 2.2.4.1. SetIniFilePath

- **Description**

This function sets the ini file path that includes the log settings.

```
void SetIniFilePath(string strIniFilePath)
```

- **Parameters**

The following table describes the parameters of the SetIniFilePath function.

In/Out	Parameter	Description
In	String	The ini file path.

**Table 19: Parameters of SetIniFilePath**

### 2.2.4.2. SetConfigurationFilePath

- **Description**

This function sets the XML UA server configuration file path.

```
void SetConfigurationFilePath(string strConfigFilePath)
```

- **Parameters**

The following table describes the parameters of the SetConfigurationFilePath function.

In/Out	Parameter	Description
In	String	The XML UA server configuration file patch.

**Table 20: Parameters of SetConfigurationFilePath**



## 2.2.5. Callbacks

### 2.2.5.1. SubscribeOnSessionCreatedEvent

- **Description**

This callback allows the server application to subscribe a function to the sessions' creation event.

```
void SubscribeOnSessionCreatedEvent(SessionsEventHandler del)
```

- **Parameters**

The following table describes the parameters of the SubscribeOnSessionCreatedEvent function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

Table 21: Parameters of SubscribeOnSessionCreatedEvent

### 2.2.5.2. UnsubscribeOnSessionCreatedEvent

- **Description**

This callback allows the server application to unsubscribe a function from the sessions' creation event.

```
void UnsubscribeOnSessionCreatedEvent(SessionsEventHandler del)
```

- **Parameters**

The following table describes the parameters of the UnsubscribeOnSessionCreatedEvent function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

Table 22: Parameters of UnsubscribeOnSessionCreatedEvent

### 2.2.5.3. SubscribeOnSessionActivatedEvent

- **Description**

This callback allows the server application to subscribe a function to the sessions' activation event.

```
void SubscribeOnSessionActivatedEvent(SessionsEventHandler del)
```

- **Parameters**

The following table describes the parameters of the `SubscribeOnSessionActivatedEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 23: Parameters of `SubscribeOnSessionActivatedEvent`**

#### 2.2.5.4. `UnsubscribeOnSessionActivatedEvent`

- **Description**

This callback allows the server application to unsubscribe a function from the sessions' activation event.

```
void UnsubscribeOnSessionActivatedEvent(SessionsEventsHandler del)
```

- **Parameters**

The following table describes the parameters of the `UnsubscribeOnSessionActivatedEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 24: Parameters of `UnsubscribeOnSessionActivatedEvent`**

#### 2.2.5.5. `SubscribeOnSessionClosedEvent`

- **Description**

This callback allows the server application to subscribe a function to the close sessions' event.

```
void SubscribeOnSessionClosedEvent(SessionsEventsHandler del)
```

- **Parameters**

The following table describes the parameters of the `SubscribeOnSessionClosedEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 25: Parameters of `SubscribeOnSessionClosedEvent`**

### 2.2.5.6. UnsubscribeOnSessionClosedEvent

- **Description**

This callback allows the server application to unsubscribe a function from the close sessions' event.

`void UnsubscribeOnSessionClosedEvent(SessionsEventsHandler del)`

- **Parameters**

The following table describes the parameters of the UnsubscribeOnSessionClosedEvent function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 26: Parameters of UnsubscribeOnSessionClosedEvent**

### 2.2.5.7. SubscribeOnSubscriptionCreatedEvent

- **Description**

This callback allows the server application to subscribe a function to the create subscriptions' event.

`void SubscribeOnSubscriptionCreatedEvent(SubscriptionsEventsHandler del)`

- **Parameters**

The following table describes the parameters of the SubscribeOnSubscriptionCreatedEvent function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 27: Parameters of SubscribeOnSubscriptionCreatedEvent**

### 2.2.5.8. UnsubscribeOnSubscriptionCreatedEvent

- **Description**

This callback allows the server application to unsubscribe a function from the create subscriptions' event.

`void UnsubscribeOnSubscriptionCreatedEvent(SubscriptionsEventsHandler del)`

- **Parameters**

The following table describes the parameters of the `UnsubscribeOnSubscriptionCreatedEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 28: Parameters of `UnsubscribeOnSubscriptionCreatedEvent`**

### 2.2.5.9. `SubscribeOnSubscriptionDeletedEvent`

- **Description**

This callback allows the server application to subscribe a function to the delete subscriptions' event.

```
void SubscribeOnSubscriptionDeletedEvent(SubscriptionsEventsHandler del)
```

- **Parameters**

The following table describes the parameters of the `SubscribeOnSubscriptionDeletedEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 29: Parameters of `SubscribeOnSubscriptionDeletedEvent`**

### 2.2.5.10. `UnsubscribeOnSubscriptionDeletedEvent`

- **Description**

This callback allows the server application to unsubscribe a function from the delete subscriptions' event.

```
void UnsubscribeOnSubscriptionDeletedEvent(SubscriptionsEventsHandler del)
```

- **Parameters**

The following table describes the parameters of the `UnsubscribeOnSubscriptionDeletedEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 30: Parameters of `UnsubscribeOnSubscriptionDeletedEvent`**

### 2.2.5.11. SubscribeOnWriteValueEvent

- **Description**

This callback allows the server application to subscribe a function to the write value event.

```
void SubscribeOnWriteValueEvent(WriteValueEventHandler del)
```

- **Parameters**

The following table describes the parameters of the SubscribeOnWriteValueEvent function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 31: Parameters of SubscribeOnWriteValueEvent**

### 2.2.5.12. UnsubscribeOnWriteValueEvent

- **Description**

This callback allows the server application to unsubscribe a function from the write value event.

```
void UnsubscribeOnWriteValueEvent(WriteValueEventHandler del)
```

- **Parameters**

The following table describes the parameters of the UnsubscribeOnWriteValueEvent function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 32: Parameters of UnsubscribeOnWriteValueEvent**

### 2.2.5.13. SubscribeOnReadRawHDAValueEvent

- **Description**

This callback allows the server application to subscribe a function to the read raw HDA value event.

```
void SubscribeOnReadRawHDAValueEvent(ReadRawHDAEventHandler del)
```

- **Parameters**

The following table describes the parameters of the SubscribeOnReadRawHDAValueEvent function.

In/Out	Parameter	Description
--------	-----------	-------------

In	Del	The delegate that will subscribe to the corresponding event.
----	-----	--

**Table 33: Parameters of SubscribeOnReadRawHDAValueEvent**

#### 2.2.5.14. SubscribeOnReadProcessedHDAValueEvent

- **Description**

This callback allows the server application to subscribe a function to the read process HDA value event.

```
void SubscribeOnReadProcessedHDAValueEvent(ReadProcessedHDAEventHandler del)
```

- **Parameters**

The following table describes the parameters of the SubscribeOnReadProcessedHDAValueEvent function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 34: Parameters of SubscribeOnReadProcessedHDAValueEvent**

#### 2.2.5.15. UnsubscribeOnReadProcessedHDAValueEvent

- **Description**

This callback allows the server application to unsubscribe a function from the read process HDA value event.

```
void UnsubscribeOnReadProcessedHDAValueEvent(ReadProcessedHDAEventHandler del)
```

- **Parameters**

The following table describes the parameters of the UnsubscribeOnReadProcessedHDAValueEvent function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 35: Parameters of UnsubscribeOnReadProcessedHDAValueEvent**

#### 2.2.5.16. SubscribeOnReadAtTimeHDAValueEvent

- **Description**

This callback allows the server application to subscribe a function to the on read at time event.

`void SubscribeOnReadAtTimeHDAValueEvent(ReadAtTimeHDAEventHandler del)`

- **Parameters**

The following table describes the parameters of the `SubscribeOnReadAtTimeHDAValueEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 36: Parameters of `SubscribeOnReadAtTimeHDAValueEvent`**

### 2.2.5.17. `UnsubscribeOnReadAtTimeHDAValueEvent`

- **Description**

This callback allows the server application to unsubscribe a function from the on read at time event.

`void UnsubscribeOnReadAtTimeHDAValueEvent(ReadAtTimeHDAEventHandler del)`

- **Parameters**

The following table describes the parameters of the `UnsubscribeOnReadAtTimeHDAValueEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 37: Parameters of `UnsubscribeOnReadAtTimeHDAValueEvent`**

### 2.2.5.18. `SubscribeOnDeleteAtTimeHDAValueEvent`

- **Description**

This callback allows the server application to subscribe a function to the on delete at time event.

`void SubscribeOnDeleteAtTimeHDAValueEvent(DeleteAtTimeHDAEventHandler del)`

- **Parameters**

The following table describes the parameters of the `SubscribeOnDeleteAtTimeHDAValueEvent` function.

In/Out	Parameter	Description
--------	-----------	-------------

In	Del	The delegate that will subscribe to the corresponding event.
----	-----	--

**Table 38: Parameters of SubscribeOnDeleteAtTimeHDAValueEvent**

### 2.2.5.19. UnsubscribeOnDeleteAtTimeHDAValueEvent

- **Description**

This callback allows the server application to unsubscribe a function from the on delete at time event.

`void UnsubscribeOnDeleteAtTimeHDAValueEvent(DeleteAtTimeHDAEventHandler del)`

- **Parameters**

The following table describes the parameters of the `UnsubscribeOnDeleteAtTimeHDAValueEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 39: Parameters of UnsubscribeOnDeleteAtTimeHDAValueEvent**

### 2.2.5.20. SubscribeOnHistoryUpdateEvent

- **Description**

This callback allows the server application to subscribe a function to the history update event.

`void SubscribeOnHistoryUpdateEvent(HistoryUpdateDataEventHandler del)`

- **Parameters**

The following table describes the parameters of the `SubscribeOnHistoryUpdateEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 40: Parameters of SubscribeOnHistoryUpdateEvent**

### 2.2.5.21. UnsubscribeOnHistoryUpdateEvent

- **Description**

This callback allows the server application to unsubscribe a function from the history update event.

`void UnsubscribeOnHistoryUpdateEvent(HistoryUpdateDataEventHandler del)`



- **Parameters**

The following table describes the parameters of the `UnsubscribeOnHistoryUpdateEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 41: Parameters of `UnsubscribeOnHistoryUpdateEvent`**

### 2.2.5.22. `SubscribeOnCertificateValidationEvent`

- **Description**

This callback allows the server application to subscribe a function to the on certificate validation event.

`void` `SubscribeOnCertificateValidationEvent(CertificateValidatorEventHandler del)`

- **Parameters**

The following table describes the parameters of the `SubscribeOnCertificateValidationEvent` function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 42: Parameters of `SubscribeOnCertificateValidationEvent`**

### 2.2.5.23. `UnsubscribeOnCertificateValidationEvent`

- **Description**

This callback allows the server application to unsubscribe a function from the on certificate validation event.

`void` `UnsubscribeOnCertificateValidationEvent(CertificateValidatorEventHandler del)`

- **Parameters**

The following table describes the parameters of `UnsubscribeOnCertificateValidationEvent` function.

In/Out	Parameter	Description
--------	-----------	-------------

In	Del	The delegate that will unsubscribe from the corresponding event.
----	-----	--

**Table 43: Parameters of UnsubscribeOnCertificateValidationEvent**

#### 2.2.5.24. SubscribeOnHistoryDeleteRaw

- **Description**

This callback allows the server application to subscribe a function to the history delete raw event.

`void` `SubscribeOnHistoryDeleteRaw`(`HistoryDeleteRawEventHandler` del)

- **Parameters**

The following table describes the parameters of the `SubscribeOnHistoryDeleteRaw` function.

In/Out	Parameter	Description
In	Del	The delegate that will subscribe to the corresponding event.

**Table 44: Parameters of SubscribeOnHistoryDeleteRaw**

#### 2.2.5.25. UnsubscribeOnHistoryDeleteRaw

- **Description**

This callback allows the server application to unsubscribe a function from the history delete raw event.

`void` `UnsubscribeOnHistoryDeleteRaw`(`HistoryDeleteRawEventHandler` del)

- **Parameters**

The following table describes the parameters of the `UnsubscribeOnHistoryDeleteRaw` function.

In/Out	Parameter	Description
In	Del	The delegate that will unsubscribe from the corresponding event.

**Table 45: Parameters of UnsubscribeOnHistoryDeleteRaw**

## 2.2.6. Certificate Management

### 2.2.6.1. TrustCertificate

- **Description**

This function trusts the certificate by adding it to the trusted certificate store of the OPC UA Server defined in its XML configuration file.

```
bool TrustCertificate(X509Certificate2 certToTrust, out string strError)
```

- **Parameters**

The following table describes the parameters of the TrustCertificate function.

In/Out	Parameter	Description
In	certifToTrust	The certificate to be trusted.
Out	strError	If the operation fails, this function will return an error message. Otherwise, it will return an empty string.

**Table 46: Parameters of TrustCertificate**

- **Returned results**

Return Code	Description
True	The operation was successful.
False	The operation failed.

**Table 47: Returned Results of TrustCertificate**

### 2.2.6.2. RejectCertificate

- **Description**

This function rejects the certificate by adding it to the rejected certificate store of the OPC UA Client defined in its XML configuration file.

```
bool RejectCertificate(X509Certificate2 certToReject, out string strError)
```

- **Parameters**

The following table describes the parameters of the RejectCertificate function.

In/Out	Parameter	Description
--------	-----------	-------------

In	certifToReject	The certificate to be rejected.
Out	strError	If the operation fails, this function will return an error message. Otherwise, it will return an empty string.

**Table 48: Parameters of RejectCertificate**

- **Returned results**

Return Code	Description
True	The operation was successful.
False	The operation failed.

**Table 49: Returned Results of RejectCertificate**

## 2.2.7. Delegates

### 2.2.7.1. SessionsEventsHandler

- **Description**

This function is used to receive session related events.

`delegate void SessionsEventsHandler(Session session, SessionEventReason reason)`

- **Parameters**

The following table describes the parameters of the SessionsEventsHandler function.

In/Out	Parameter	Description
In	Session	The session that was affected.
In	Reason	The reason.

*Table 50: Parameters of SessionsEventsHandler*

### 2.2.7.2. SubscriptionsEventsHandler

- **Description**

This function is used to receive subscription related events.

`delegate void SubscriptionsEventsHandler(Subscription subscription, bool deleted)`

- **Parameters**

The following table describes the parameters of the SubscriptionsEventsHandler function.

In/Out	Parameter	Description
In	subscription	The subscription concerned by the event.
In	Deleted	True if the subscription was deleted.

*Table 51: Parameters of SubscriptionsEventsHandler*

### 2.2.7.3. WriteValueEventHandler

- **Description**

This function is used to report the results of a write value operation.

`delegate ServiceResult WriteValueEventHandler(ISystemContext context, NodeState node, NumericRange indexRange, QualifiedName dataEncoding, ref object value, ref StatusCode statusCode, ref DateTime timestamp)`

- **Parameters**

The following table describes the parameters of the WriteValueEventHandler function.

In/Out	Parameter	Description
In	Context	The request context.
In	Node	The node.
In	indexRange	The index range.
In	dataEncoding	The data encoding.
In	Value	The value.
In	statusCode	The code status.
In	Timestamp	The time stamp.

**Table 52: Parameters of WriteValueEventHandler**

- **Returned results**

Return Code	Description
Good	The operation was successful.
Bad	The operation failed.
Uncertain	The operation failed but no specific reason is known.

**Table 53: Returned Results of WriteValueEventHandler**

#### 2.2.7.4. ReadRawHDAEventHandler

- **Description**

This function is used to report the results of a history read raw operation.

`delegate void ReadRawHDAEventHandler(ReadRawModifiedDetails`

`ReadRawModifiedDetails, NodeHandle handle, out List<ReadHDAValuesResult> Results)`

- **Parameters**

The following table describes the parameters of the ReadRawHDAEventHandler function.

In/Out	Parameter	Description
--------	-----------	-------------

In	ReadRawModifiedDetails	The details concerning the history read raw operation.
In	Handle	The node handle
Out	Results	The list of returnedHDA values.

**Table 54: Parameters of ReadRawHDAEventHandler**

### 2.2.7.5. ReadProcessedHDAEventHandler

- **Description**

This function is used to report the results of a history read process operation.

`delegate void ReadProcessedHDAEventHandler(ReadProcessedDetails details, NodeHandle handle, out List<ReadHDAValuesResult> Results)`

- **Parameters**

The following table describes the parameters of the ReadProcessedHDAEventHandler function.

In/Out	Parameter	Description
In	ReadProcessedDetails	The details concerning the history read process operation.
In	Handle	The node handle.
Out	Results	The list of returned HDA values.

**Table 55: Parameters of ReadProcessedHDAEventHandler**

### 2.2.7.6. ReadAtTimeHDAEventHandler

- **Description**

This function is used to report the results of a history read at time operation .

`delegate void ReadAtTimeHDAEventHandler(ReadAtTimeDetails details, NodeHandle handle, out List<ReadHDAValuesResult> Results)`

- **Parameters**

The following table describes the parameters of the ReadAtTimeHDAEventHandler function.

In/Out	Parameter	Description
--------	-----------	-------------

In	Details	The details concerning the read at time operation
In	Handle	The node handle.
Out	Results	The list of returned HDA values.

**Table 56: Parameters of ReadAtTimeHDAEventHandler**

### 2.2.7.7. DeleteAtTimeHDAEventHandler

- **Description**

This function is used to report the results of a history delete at time operation .

```
delegate void DeleteAtTimeHDAEventHandler(IList<DeleteAtTimeDetails> details,
                                         NodeHandle handle)
```

- **Parameters**

The following table describes the parameters of the DeleteAtTimeHDAEventHandler function.

In/Out	Parameter	Description
In	Details	The details concerning the read At time operation
In	Handle	The node handle.

**Table 57: Parameters of DeleteAtTimeHDAEventHandler**

### 2.2.7.8. CallMethodEventHandler

- **Description**

This delegate is called to invoke the method.

```
delegate void CallMethodEventHandler(IList<object> inputArguments, out VariantCollection
                                     outputArguments)
```

- **Parameters**

The following table describes the parameters of the CallMethodEventHandler function.

In/Out	Parameter	Description
In	inputArguments	The input arguments
Out	VariantCollection outputArgument	The output arguments.

**Table 58: Parameters of CallMethodEventHandler**



### 2.2.7.9. CertificateValidatorEventHandler

- **Description**

This function is used to validate a certificate operation.

```
delegate void CertificateValidatorEventHandler(CertificateValidator validator,
                                             CertificateValidationEventArgs e)
```

- **Parameters**

The following table describes the parameters of the CertificateValidatorEventHandler function.

In/Out	Parameter	Description
In	Validator	The validator.
In	e	The event.

**Table 59: Parameters of CertificateValidatorEventHandler**

### 2.2.7.10. HistoryUpdateDataEventHandler

- **Description**

This function is used to report the results of a history update data operation.

```
delegate void HistoryUpdateDataEventHandler(UpdateDataDetails DataToBeUpdated,
                                             NodeHandle handle)
```

- **Parameters**

The following table describes the parameters of the HistoryUpdateDataEventHandler function.

In/Out	Parameter	Description
In	DataToBeUpdated	The details concerning the history update data
In	Handle	The node handle.

**Table 60: Parameters of HistoryUpdateDataEventHandler**

### 2.2.7.11. HistoryDeleteRawEventHandler

- **Description**

This function is used to report the results of a history delete raw operation.

```
delegate void HistoryDeleteRawEventHandler(DeleteRawModifiedDetails details, NodeHandle
                                             handle)
```

- **Parameters**

The following table describes the parameters of the HistoryDeleteRawEventHandler function.

In/Out	Parameter	Description
In	Details	The details concerning the history delete raw operation.
In	Handle	The node handle.

**Table 61: Parameters of HistoryDeleteRawEventHandler**

## 3. Folder Class

### 3.1. Constructor

- **Description**

This constructor defines a new object of the Folder class.

```
public Folder (string Path, string Name)
```

- **Parameters**

The following table describes the parameters of the Folder class:

In/Out	Parameter	Description
In	Path	The path to the folder.
In	Name	The name of the folder.

**Table 62: Parameters of Folder Constructor**

## 4. Variable Class

### 4.1. Constructor

- **Description**

This constructor defines a new object of the Variable class.

```
public Variable (string Path, string Name, NodeId DataType, int ValueRank)
```

- **Parameters**

The following table describes the parameters of the Variable constructor:

In/Out	Parameter	Description
In	Path	The path to the variable.
In	Name	The name of the variable.
In	DataType	The data type of the node identifier.
In	ValueRank	Indicates whether the DataType is an array and how many dimensions the array has.

**Table 63: Parameters of Variable Constructor**

## 4.2. Functions

- **Description**

This function updates the variable items.

`bool` UpdateItem (UAServerManager manager, `object` value, `DateTime` TimeStamp, `StatusCode` statusCode, `out string` strError)

- **Parameters**

The following table describes the parameters of the UpdateItem function.

In/Out	Parameter	Description
In	Manager	The UA server manager to be updated.
In	Value	The object value to be updated.
In	TimeStamp	The time stamp to be updated.
In	statusCode	The status code to be updated.
Out	strError	If the operation fails, this function will return an error message. Otherwise, it will return an empty string

**Table 64: Parameters of UpdateItem**

- **Returned results**

Return Code	Description
True	The operation was successful.
False	The operation failed.

**Table 65: Returned Results of UpdateItem**

## 5. Method Class

### 5.1. Constructor

- **Description**

This constructor defines a new object of the Method class.

```
public Method(string path, string name, List<Argument> lstInputArguments, List<Argument>
    lstOutputArguments, CallMethodEventHandler delMethodCall)
```

- **Parameters**

The following table describes the parameters of the Method constructor.

In/Out	Parameter	Description
In	Path	The path to the method.
In	Name	The name of the method.
In	IstInputArguments	List of input argument.
In	IstOutputArguments	List of output argument.
In	delMethodCall	The delegate of the call method event handler.

**Table 66: Parameters of Method Constructor**

### 5.2. Functions

#### 5.2.1. ExecuteMethod

- **Description**

This function executes the methods.

```
void ExecuteMethod (IList<object> inputArguments, out VariantCollection outputArguments)
```

- **Parameters**

The following table describes the parameters of the UpdateItem function.

In/Out	Parameter	Description
In	inputArguments	The list of input arguments of the method

Out	outputArguments	The list of output arguments of the method
-----	-----------------	--

**Table 67: Parameters of UpdateItem**

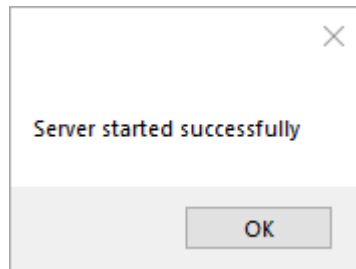
# USING OPC UA SAMPLE SERVERS

This chapter describes the steps on how to use the OPC UA sample servers available within the OPC UA Server Toolkit installation folder:

- OPC UA Sample Server C#, which is a GUI application developed in C# .NET, provided in both 32 and 64 bit versions.
- OPC UA Sample Server VB, which is a Windows service sample developed in VB .NET, provided in both 32 and 64 bit versions.

## 1. OPC UA Sample Server C#

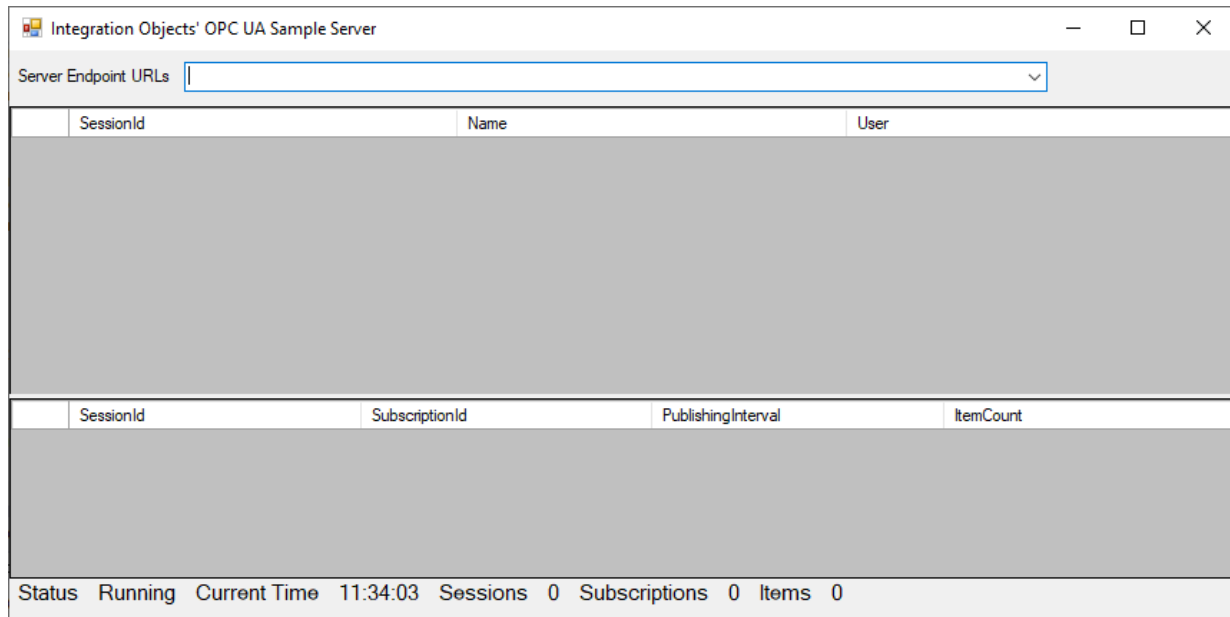
Run the OPC UA Sample Server executable using an administrator account and you will get a message indicating that the server started successfully if your license is valid.



**Figure 19: Run the OPC UA Sample Server**

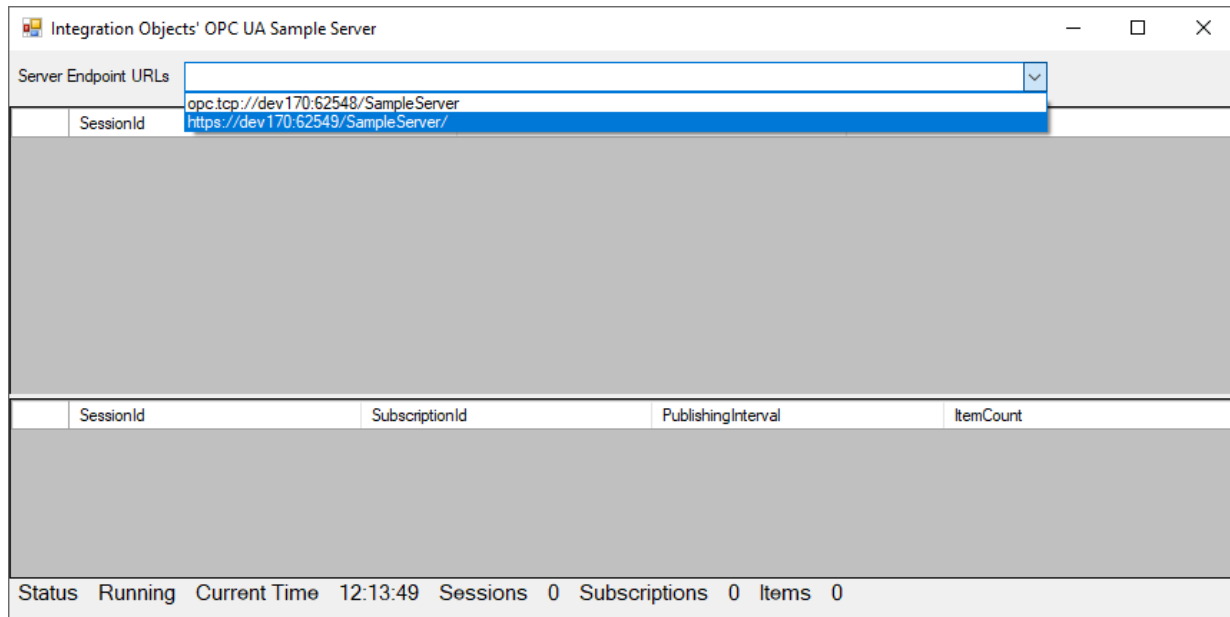
The OPC UA Server user interface will then be displayed. This interfaces displays:

- The server endpoint URLs,
- The number of sessions currently open,
- The number of subscriptions currently established
- The current status (running, shutdown etc.)
- And the current time and the total number of sessions, subscriptions, and displayed items.



**Figure 20: OPC UA Sample Server User Interface**

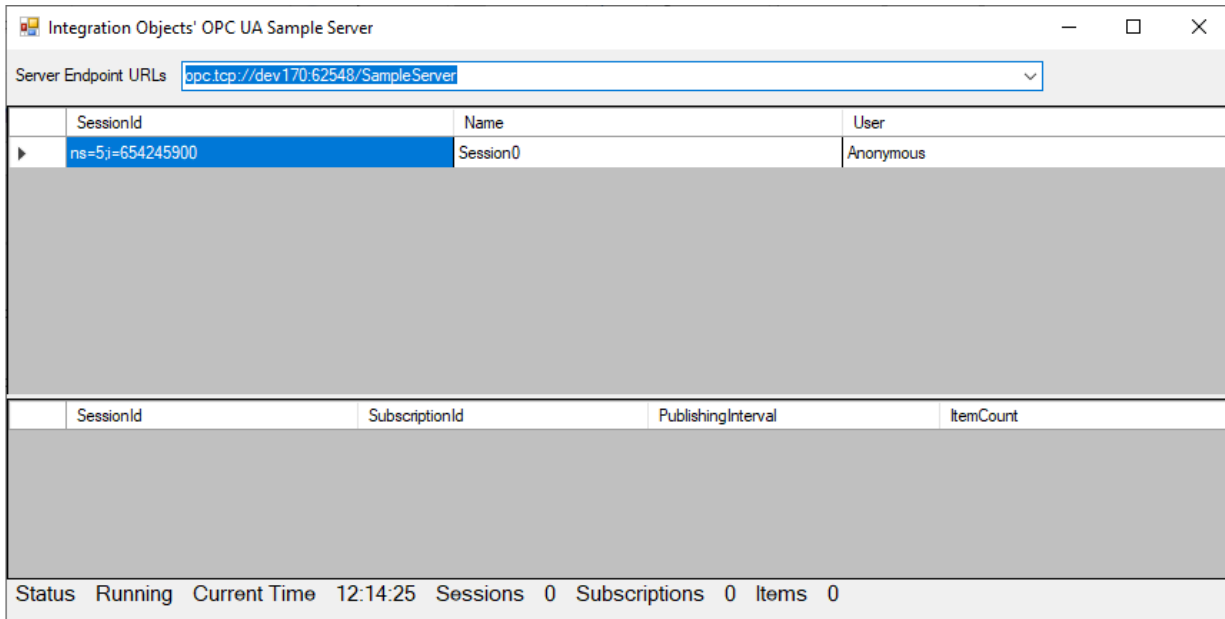
The OPC UA Sample Server can be accessed on the two endpoints that are available from the Server Endpoint URLs dropdown list, as illustrated below:



**Figure 21: OPC UA Server Endpoints URLs**

Simply copy one of the server endpoints and then paste it into an OPC UA Client to establish a connection with the OPC UA Sample Server. Once you are connected, a session is opened. This session provides the related state information: sessionId, name and user.





The screenshot shows the 'Integration Objects' OPC UA Sample Server window. The 'Server Endpoint URLs' field is set to 'opc.tcp://dev170:62548/SampleServer'. Below this, there are two tables. The first table shows a session with SessionId 'ns=5:j=654245900', Name 'Session0', and User 'Anonymous'. The second table shows subscription information with columns for SessionId, SubscriptionId, PublishingInterval, and ItemCount. The status bar at the bottom indicates 'Status Running', 'Current Time 12:14:25', 'Sessions 0', 'Subscriptions 0', and 'Items 0'.

SessionId	Name	User
ns=5:j=654245900	Session0	Anonymous

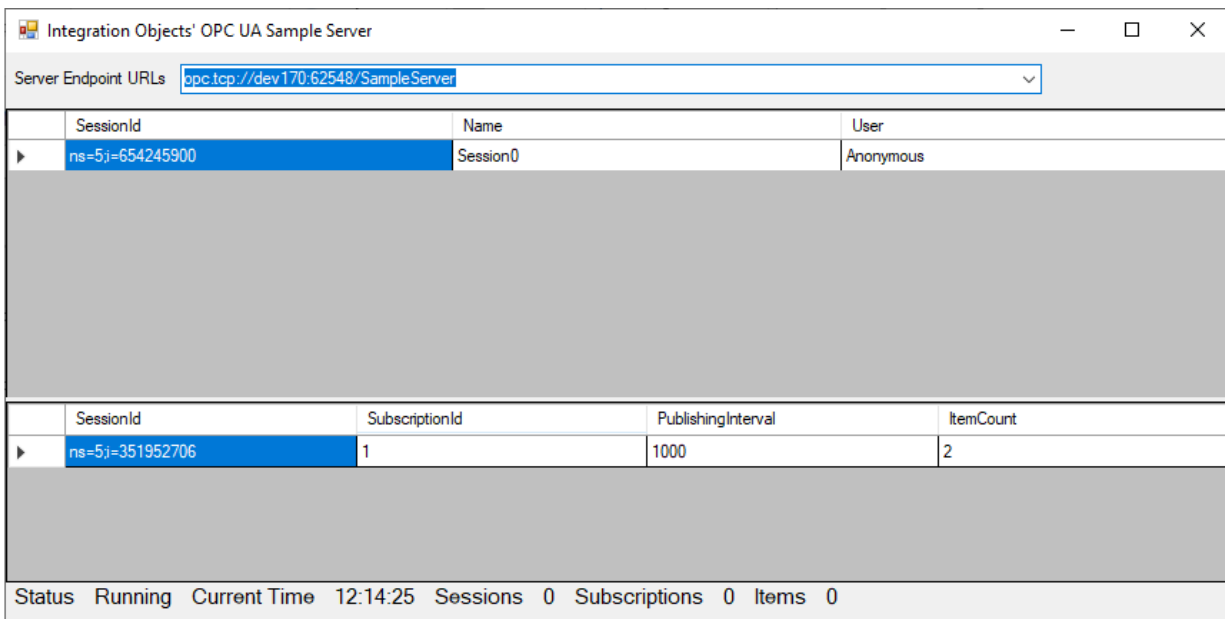
  

SessionId	SubscriptionId	PublishingInterval	ItemCount

Status Running Current Time 12:14:25 Sessions 0 Subscriptions 0 Items 0

**Figure 22: Connection Established from an OPC UA Client**

Then, use your OPC UA client to establish a subscription with the server. The server provides the related state information of the subscription as shown in the figure below.



The screenshot shows the 'Integration Objects' OPC UA Sample Server window. The 'Server Endpoint URLs' field is set to 'opc.tcp://dev170:62548/SampleServer'. Below this, there are two tables. The first table shows a session with SessionId 'ns=5:j=654245900', Name 'Session0', and User 'Anonymous'. The second table shows subscription information with columns for SessionId, SubscriptionId, PublishingInterval, and ItemCount. The status bar at the bottom indicates 'Status Running', 'Current Time 12:14:25', 'Sessions 0', 'Subscriptions 0', and 'Items 0'.

SessionId	Name	User
ns=5:j=654245900	Session0	Anonymous

SessionId	SubscriptionId	PublishingInterval	ItemCount
ns=5:j=351952706	1	1000	2

Status Running Current Time 12:14:25 Sessions 0 Subscriptions 0 Items 0

**Figure 23: Subscriptions and Monitored Items**

## 2. OPC UA Sample Service VB

With the OPC UA Server Toolkit installation, an OPC UA Sample service will be installed automatically according to your machine architecture (x86 or x64)

To run the OPC UA Sample Server Service:

- Open Task manager
- Go to Services
- Then, locate the “Integration Objects’ OPC UA Sample Server Service” service and right click on it
- Select start from the displayed menu

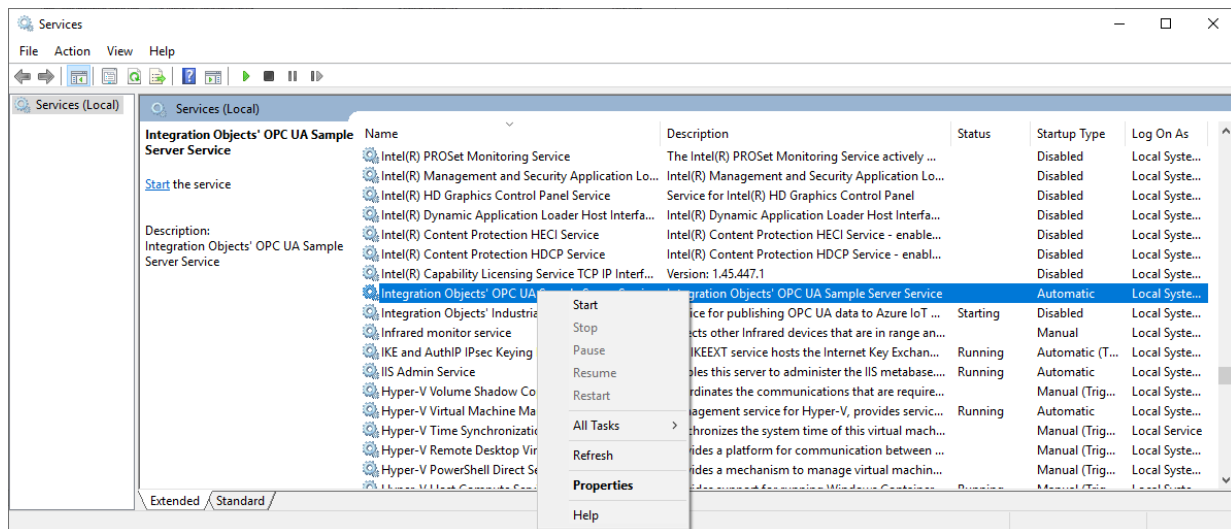


Figure 24: Start the OPC UA Sample Server Service

You can locate the Endpoints URLs of the OPC UA Sample Server Service in the “SampleServer.config” file

```

<?xml version="1.0" encoding="utf-8"?>
<ApplicationConfiguration
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ua="http://opcfoundation.org/UA/2008/02/Types.xsd"
  xmlns="http://opcfoundation.org/UA/SDK/Configuration.xsd"
>
  <ApplicationName>Integration Objects Sample Server</ApplicationName>
  <ApplicationUri>urn:localhost:UA:SampleServer</ApplicationUri>
  <ProductUri>http://integrationobjects.com/SampleServer</ProductUri>
  <ApplicationType>Server_0</ApplicationType>

  <SecurityConfiguration>
    <TransportConfigurations></TransportConfigurations>
    <TransportQuotas>
      <ServerConfiguration>
        <BaseAddresses>
          <ua:String>https://localhost:62549/SampleServer</ua:String>
          <ua:String>opc.tcp://localhost:62548/SampleServer</ua:String>
        </BaseAddresses>
        <SecurityPolicies>
          <ServerSecurityPolicy>
            <ServerSecurityPolicy>
              <ServerSecurityPolicy>
                <SecurityMode>SignAndEncrypt_3</SecurityMode>
                <SecurityPolicyUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</SecurityPolicyUri>
              </ServerSecurityPolicy>
            </ServerSecurityPolicy>
          </SecurityPolicies>
        </ServerConfiguration>
      </TransportQuotas>
    </SecurityConfiguration>
  </ApplicationConfiguration>

```

**Figure 25 : XML Configuration File**

Copy one of the Endpoint URLs and then paste it into your OPC UA Client to establish a connection with the OPC UA Sample Server Service.

# TOOLKIT TRACING CAPABILITIES

The toolkit provides tracing capabilities. Developers can record the toolkit errors and debugging information in a log file named **OPCUAServerToolkitLog.LOG**. If difficulties occur with the toolkit, the log file can be extremely valuable for troubleshooting.

This log file is generated at start-up where the server executable file is located. The toolkit incorporates a configuration file `Config.ini` that includes several logging parameters. All these parameters have default settings and can be changed at start-up by editing the configuration file.

To change this configuration file:

1. Open the `Config.ini` file using a text editor.
2. Edit any of the parameters listed in the following tables:

Log Setting	Description	Default Value
<b>AutoAppend</b>	Set to true to continue writing log messages in the original log file or to false to create a new file.	True
<b>BufferSize</b>	The maximum number of messages to be stored in the runtime memory before launching a write action into the hard disk. The specified value must be greater than 100.	100
<b>LogFileMaxSize</b>	The maximum size of the log file	10 MB
<b>MaximumFiles</b>	Set to 0 means that log files will be created in an unlimited way.	0
<b>AutoSaveTimeOut</b>	The time to wait to read all messages from the buffer and write it on hard disk, the minimum value is 10s.	10
<b>Level</b>	There are five log levels: <ol style="list-style-type: none"> <li>1. <b>Control</b>: Logs only control messages generated by the toolkit.</li> <li>2. <b>Error</b>: Logs error and control messages generated by the toolkit.</li> </ol>	Error

	<ol style="list-style-type: none"> <li>3. <b>Warning:</b> Logs warning, error and control messages generated by toolkit.</li> <li>4. <b>Inform:</b> Logs information, warning, error and control messages generated by the toolkit.</li> <li>5. <b>Debug:</b> Logs all messages generated by the toolkit.</li> </ol> <p>The higher the log level, the more information are recorded. We recommend using level “Control” for a better performance of the application. The other levels are dedicated for troubleshooting purposes.</p>	
--	---	--

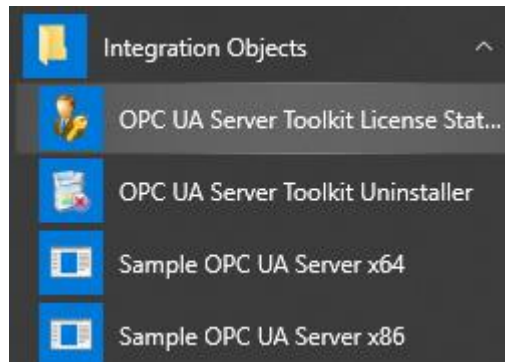
**Table 68: Log Settings**

3. Save the file and restart your server application for the log settings for changes to take effect.

# TROUBLESHOOTING

## Case 1: Unable to register the UA Server

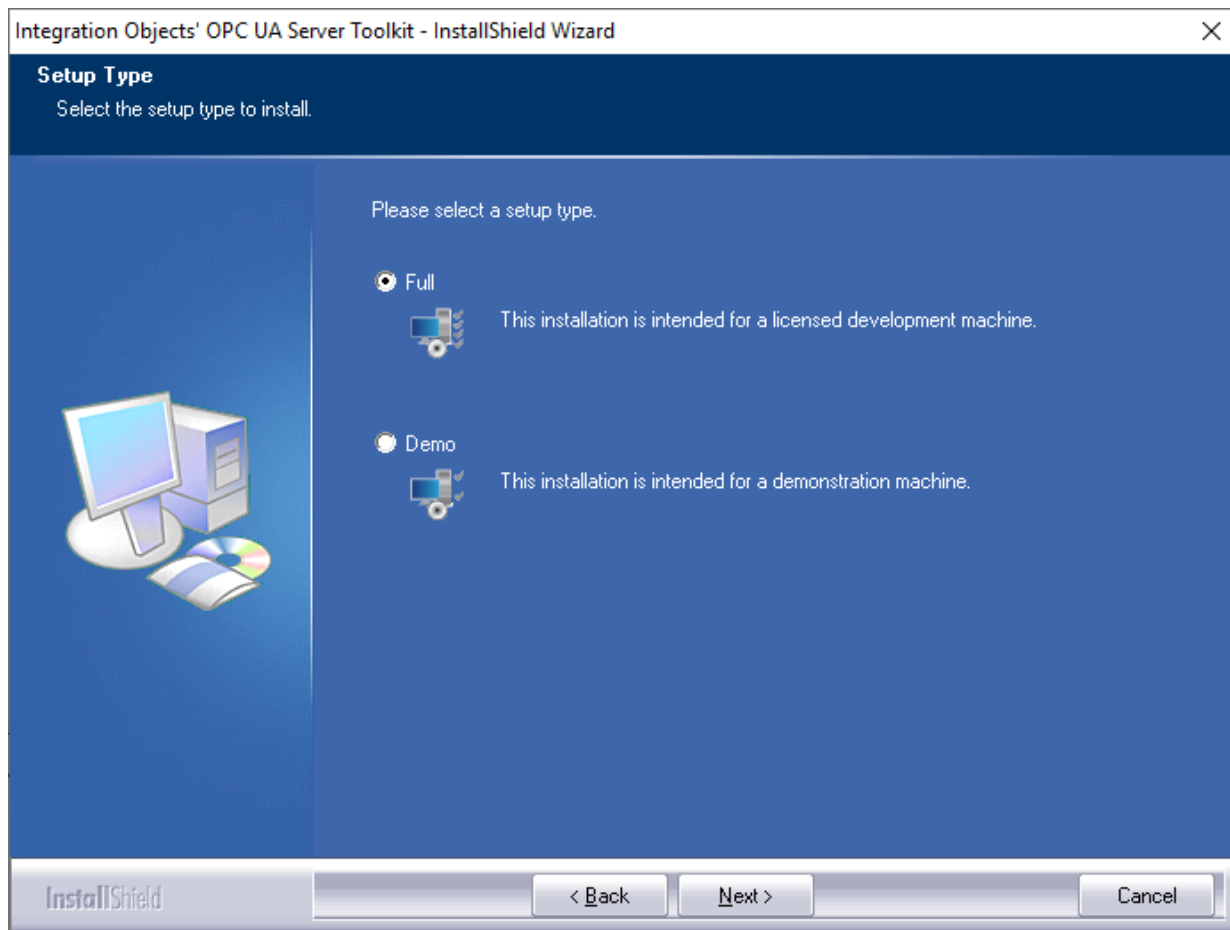
If you are using an evaluation license, first check the license validity using the License Status tool. You can start this tool from the startup menu as illustrated below:



**Figure 26: OPC UA Server Toolkit Start Menu**

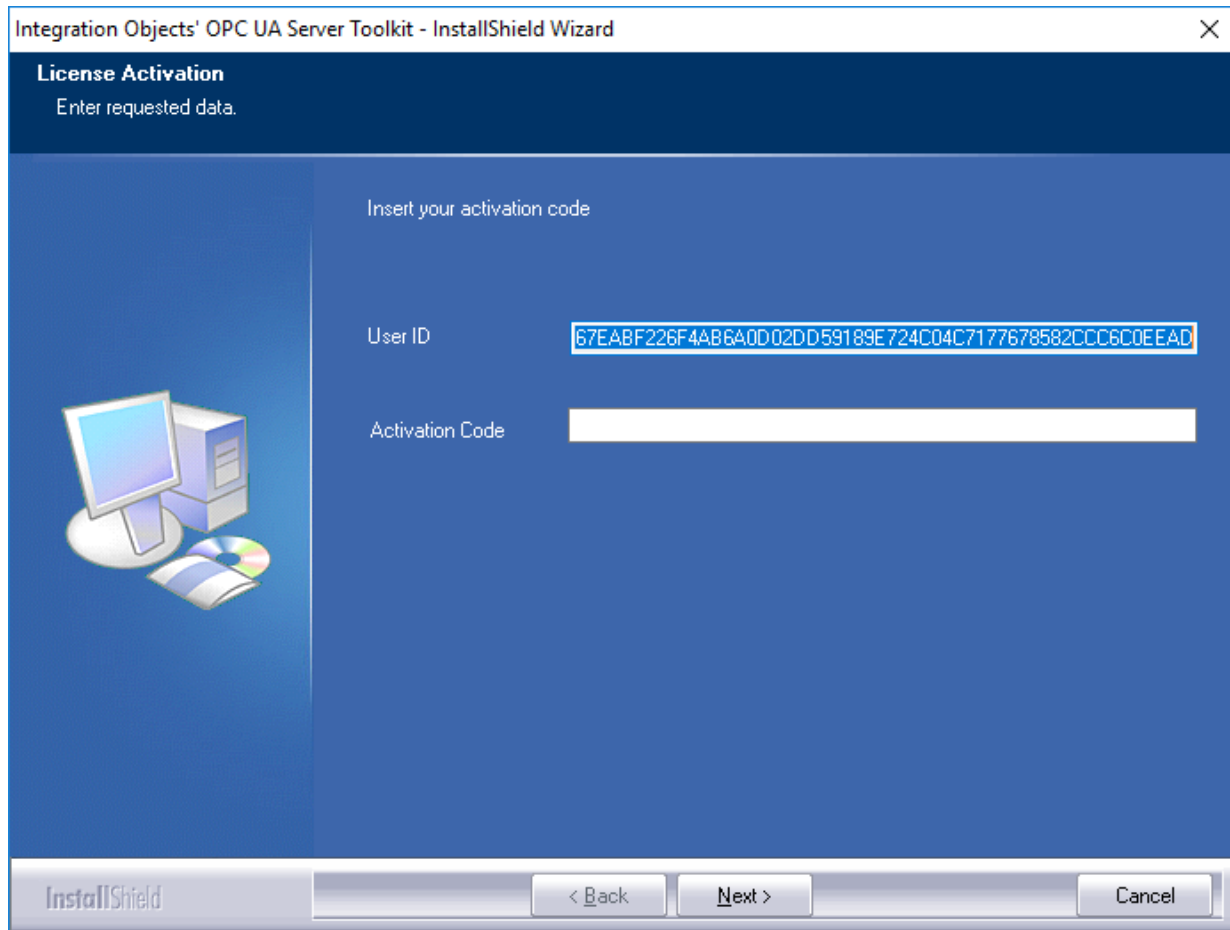
If the License Status tool shows that the demo expired and you want to activate your license using an already purchased full license, follow the steps below:

1. Uninstall the OPC UA Server Toolkit.
2. Install OPC UA Server Toolkit using an administrator account in your licensed development machine.
3. When prompted, select the “Full version” option in the Setup Type dialog.



**Figure 27: Select Type Dialog**

4. Copy and send the User ID to the sales team {[sales@integrationobjects.com](mailto:sales@integrationobjects.com)} so they can generate the activation code for you.



**Figure 28: License Activation Dialog**

5. Enter the received code in the **Activation Code** field and click the **Next** button.

### Case 2: “This is not a development machine” Error Message

Given that the OPC UA Server Toolkit is licensed per development machine, make sure you are developing your application in the machine having the OPC UA Server Toolkit installed otherwise you will be encountered with the following error message in the log file : **“This is not a development machine”**.

### Case 3: “This is not a valid license” Error Message

When you run your server application, the following error message is displayed in the log file : **“This is not a valid license”**.



Open the license status tool and check the license status to verify that the license is valid.

Else, If you are running your application using a full runtime version of the OPC UA Server Toolkit, make sure that the OPC UA Server Toolkit is not installed as a demo version in the deployment machine. If it is the case, you will need to uninstall it.

Also make sure the application deployment folder contains the following files:

- The application executable and any other custom depending assembly.
- The UA XML configuration file (XXXX.Config.xml, where XXXX is the name of your OPC UA server application).
- Config.ini.
- IntegrationObjects.Logger.SDK.dll.
- IntegrationObjects.Opc.Ua.Server.Toolkit.dll.
- IntegrationObjects.Opc.Ua.Server.dll.
- IntegrationObjects.Opc.Ua.Core.dll.
- IntegrationObjects.Opc.Ua.Configuration.dll.
- IntegrationObjects.Opc.Ua.Utilities.dll
- libuv.dll
- License.dll.
- libuv.dllBouncyCastle.Crypto.dll.
- Microsoft.AspNetCore.Hosting.Abstractions.dll.
- Microsoft.AspNetCore.Hosting.dll.
- Microsoft.AspNetCore.Hosting.Server.Abstractions.dll.
- Microsoft.AspNetCore.Http.Abstractions.dll.
- Microsoft.AspNetCore.Http.dll.
- Microsoft.AspNetCore.Http.Features.dll.
- Microsoft.AspNetCore.Server.Kestrel.dll.
- Microsoft.AspNetCore.Server.Kestrel.Https.dll.
- Microsoft.AspNetCore.WebUtilities.dll.
- Microsoft.Extensions.Configuration.Abstractions.dll.
- Microsoft.Extensions.Configuration.dll.
- Microsoft.Extensions.Configuration.EnvironmentVariables.dll.

- Microsoft.Extensions.DependencyInjection.Abstractions.dll.
- Microsoft.Extensions.DependencyInjection.dll.
- Microsoft.Extensions.FileProviders.Abstractions.dll.
- Microsoft.Extensions.FileProviders.Physical.dll.
- Microsoft.Extensions.FileSystemGlobbing.dll.
- Microsoft.Extensions.Logging.Abstractions.dll.
- Microsoft.Extensions.Logging.dll.
- Microsoft.Extensions.ObjectPool.dll.
- Microsoft.Extensions.Options.dll.
- Microsoft.Extensions.PlatformAbstractions.dll.
- Microsoft.Extensions.Primitives.dll.
- Microsoft.Net.Http.Headers.dll.
- Newtonsoft.Json.dll.
- System.Buffers.dll.
- System.Collections.Immutable.dll.
- System.Diagnostics.DiagnosticSource.dll.
- System.Numerics.Vectors.dll.
- System.Reflection.Metadata.dll.
- System.Runtime.CompilerServices.Unsafe.dll.
- System.Runtime.InteropServices.RuntimeInformation.dll.
- System.Text.Encodings.Web.dll.
- System.Threading.Tasks.Extensions.dll.

#### **Case 4: I Sent the User ID to Integration Objects. Can I Close the Setup Program Now?**

You can close the setup program. The user ID will not change the next time you run the setup. Once you receive the activation code, go through the installation procedure using an administrator account and enter the provided code.

#### **Case 5: Do I Need to Buy a Third Party Library to Be Able to Use This Toolkit?**

No. The only license to be activated is the OPC UA Server Toolkit license in your development machine.

**Case 6: By Purchasing the Rights to the OPC UA Server Toolkit, Are We Entitled to Install the Library Only on 1 Machine?**

The OPC UA Server Toolkit is licensed per development machine. Meaning, one license can be installed on a single development machine. With respect to runtime, you can deliver as many as you want for free.

**Case 7: Is it Possible to Integrate the Library with Windows Service?**

Yes, you can use the OPC UA Server Toolkit to develop your application as Windows service.

**Case 8: Does the OPC UA Server Toolkit Support 64-bit?**

The OPC UA Server Toolkit supports 64 bit and 32 bit applications.

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

**Offices**

- Americas: +1 713 609 9208
- Europe-Africa-Middle East: +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 our website [www.integrationobjects.com](http://www.integrationobjects.com)