

Migration Guide to OPC UA .NET Standard SDK 3.50

by Softing Industrial Automation GmbH

*This is the migration guide for OPC UA .NET SDK to the new
OPC UA .NET Standard SDK.*

Table of Contents

Part I Welcome	1
1 What's New	3
2 Revision History	4
3 System Requirements	16
4 OPC UA Supported Features	18
Part II Client API Migration Guide	19
Part III Reduce Client's Output Folder Size	34
Part IV Server API Migration Guide	35
Part V Reduce Server's Output Folder Size	39
Part VI Nuget Packages	40
Part VII Copyrights	42
Index	0

1 Welcome



optimize!
softing

The OPC UA .NET Standard SDK enables fast integration of the OPC Unified Architecture standard communication interface within end user applications. It provides a comprehensive set of source code and binaries covering OPC UA functionality for building Client respective Server interface thus allowing short time to market of your solutions.

The SDK's libraries are accompanied with an easy to use API, allowing the user to concentrate on the own business requirements implementation of his product and hiding the OPC UA protocol complexity.

- **Leading edge easy-to-use client interface**
- **Suitable for a wide range of applications**
- **Higher portability**
- **Validated by solid, well established battery of tests**

The OPC UA .NET Standard SDK includes:

- OPC UA .NET Standard SDK Client Library & OPC UA .NET Standard SDK Server Library - an easy to use API built on top of the OPC UA .NET Standard Stack from OPC Foundation.
- Unchanged access to OPC UA .NET Standard Stack from OPC Foundation.
- Reworked, comprehensive documentation focusing on the user's needs on different level of OPC UA expertise.
- Support for custom complex data types decoding, compliant with OPC UA 1.03 specification.
- Sample server and client applications.

The reader is expected to be familiar with Microsoft .NET and C# terminology as well as the basic OPC UA concepts. However the documentation will try to fill some gaps between the UA concepts and the way they are realized within the source code.

This document does not describe every class in the OPC UA .NET Standard Development Kit. It is intended to supplement the Visual Studio IntelliSense documentation and help developers understand how to develop production servers and clients.

1.1 What's New

A list of the new features added in the released versions is provided below:

Version 3.50

- Integrates the new version of **OPC UA .NET Standard Stack from OPC Foundation (NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96)** which includes:
- Fixes for the following cybersecurity vulnerabilities: CVE-2023-27321, CVE-2023-31048.
- Improve reverse connect functionality.
- Add partial support for asynchronous service calls.
- Sample code is available on [github](#).

- OPC UA .NET Standard SDK Server Library:

- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- Improve reverse connect functionality into a network environment and introduce a callback for a longer period detection.
- Bug fixes.

Note: SDK libraries are compiled using **NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96**. When using OPC UA .NET Standard SDK version 3.50 *it is mandatory to use the same version of OPC UA .NET Standard Stack from OPC Foundation.*

1.2 Revision History

☒ Version 3.40

A list of the new features added in the released versions is provided below:

- Integrates the new version of **OPC UA .NET Standard Stack from OPC Foundation(NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.60)** which includes:
 - Implement a prevention mechanism against brute forcing passwords.
 - Improve the connect timeout of the first session created.
 - Sample code is available on [github](#).

- OPC UA .NET Standard SDK Server Library:

- The server library offers the possibility to configure a mechanism for stopping brute force client attempts to log in with invalid credentials.
- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- Bug fixes.

Note: SDK libraries are compiled using **NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.60**. When using OPC UA .NET Standard SDK version 3.50 *it is mandatory to use the same version of OPC UA .NET Standard Stack from OPC Foundation*.

☒ Version 3.30

- Integrates the new version of **OPC UA .NET Standard Stack from OPC Foundation(NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.370.12)** which includes:
 - The fix for the following cybersecurity vulnerability: CVE-2022-33916.
As a consequence of this fix the SampleServer DiagnosticsSummary information was changed.
In order to be able to see the entire information from the ServerDiagnosticsSummary node (the SubscriptionDiagnosticsArray and all the SessionDiagnosticsObject instances) the current session must have SysAdmin rights and use an encrypted channel.
The client must connect with user: admin, password: admin which has sysadmin privileges and use a secure encrypted channel to be able to see all diagnostics information.
All other connections are considered "Unauthorized" and can examine only their session related information as written in the specification.
- OPC UA Server Compliance Certification. (CTT Version: 1.04.9.398)
- Sample code is available on [github](#).

- OPC UA .NET Standard SDK Server Library:

- Implementation of Audit Events for all available services.

- SampleClient and SampleServer alarms changes.
- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- Bug fixes.

Note: SDK libraries are compiled using **NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.370.12**. When using OPC UA .NET Standard SDK version 3.50 ***it is mandatory to use the same version of OPC UA .NET Standard Stack from OPC Foundation.***

☒ Version 3.20

- Integrates the new version of **OPC UA .NET Standard Stack from OPC Foundation (NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.369.30)** which includes:
 - The fixes for the following cybersecurity vulnerabilities: CVE-2022-29862, CVE-2022-29863, CVE-2022-29864, CVE-2022-29865, CVE-2022-29866.
 - Improvements for Tracing and Logging described in [The logging solution in Version 1.4.368](#).
- Added support for Windows 11 and Windows Server 2022.
- Removed support for .NET Core 2.1 SDK.
- Add support for Visual Studio 2022.
- HTTPS support of OPC UA .NET Standard Stack from OPC Foundation was tested and added as supported by the OPC UA .NET Standard SDK.
- Sample code is available on [github](#).

- OPC UA .NET Standard SDK Server Library:

- The PubSub node manager from SampleServer was disabled by default. It can be easily enabled by uncommenting a line of code.
- More alarm types were added to the SampleServer.
- Added sample implementation of a custom role (see Define And Use Custom Role)
- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- Added method UaApplication.CreateSessionCopy that will create a new session based on an existing connected session and it will also copy the custom data type information from the original object.
- New feature added for reusing the custom data type information loaded into a ClientSession across the entire life of that ClientSession. See ReuseCustomTypeInfoAtReconnect section for more details.
- The connect logic from ClientSession was reworked. ConnectAsync and Connect methods will return only after the custom data types are loaded as configured by **DecodeCustomDataTypes** and **DecodeDataTypeDictionaries** flags from ClientToolkitConfiguration. There is no need to wait until the flags that indicate if custom data types are loaded become true.

The methods **FetchDataDictionaries()** and **FetchDataTypeDefinitions()** and all their variations were marked as obsolete since it is not recommended to be called from an already connected session while the session is receiving data change notifications.

- Bug fixes.

Note: SDK libraries are compiled using **NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.369.30**. When using OPC UA .NET Standard SDK version 3.20 **it is mandatory to use the same version of OPC UA .NET Standard Stack from OPC Foundation**.

☒ Version 3.10

• - OPC UA .NET Standard SDK Server Library:

- Handling of Variable node default values changed. When a new variable node is created using the Create methods from NodeManager, its value is not automatically initialized, it remains *null*, and the status code of the node is *BadWaitingForInitialData*. The first value assignment will reset the variable status code to *Good*.
- Bug fixes:
 - The *CreateObjectFromType* method from *NodeManager* will preserve the *DataType* and all other properties of the child nodes that created based on object type definition.
 - The Help file is updated with information regarding the *OperationLimits*. The *OperationLimits* settings were added to SampleServer config file.

- OPC UA .NET Standard SDK Client Library:

- Bug fixes:
 - The loading of custom data types that did not end was fixed.

SDK libraries are compiled using **OPCFoundation.NetStandard.Opc.Ua - version 1.4.367.100**. When using OPC UA .NET Standard SDK version 3.10 **it is mandatory to use the same version of OPC UA .NET Standard Stack from OPC Foundation**.

☒ Version 3.00

- The changes in **PubSub Library** from OPC UA .NET Standard Stack from OPC Foundation include *Discovery Messages for JSON and UADP* encodings and the implementation of *KeyFrameCount* that allows sending delta frames not only key frames for both: JSON and UADP encodings.
- The OPC UA .NET Standard SDK is also deployed as nuget packages on <https://nuget.org>. See section [Nuget Packages](#) for more details.

- OPC UA .NET Standard SDK Server Library:

- **Breaking change:**

- The *IEncodeableFactory* extension methods from OPC UA .NET Standard SDK Server Library have a new parameter of type *Softing.Opc.Ua.Server.Types.Domain*. This parameter allows the usage of both Server and Client libraries in the same application without custom data type implementation interfering from one domain to the other.

The methods that have this new parameter are: *GetComplexTypeInfo()*, *GetTypeTable()*.

- The *ServerCertificateLifeTime* property was added to *ServerToolkitConfiguration* to allow creating the server certificate with a custom lifetime.
- *NodeManager* class has two new protected methods that can be useful in creating unique *NodeIds* that have an *uint Identifier*:
 - *GetLastUsedNodeIdIdentifier(ushort namespaceIndex)*
 - *SetLastUsedNodeIdIdentifier(ushort namespaceIndex, uint lastUsedIdentifier)*
- Bug fixes:
 - The *CreateObjectFromType* method from *NodeManager* will preserve the *NamespaceIndex* of the *BrowseNames* for the child nodes that created based on object type definition. However, the node returned by this method will have the *NamespaceIndex* of current node manager.
 - *CreateObjectFromType* was modified to allocate only the needed node ids and this may lead to different values for *NodeId.Identifier* if the node ids were generated using the default *New()* method from *NodeManager* class.

- OPC UA .NET Standard SDK Client Library:

- **Breaking changes:**

- The API changes from OPC UA .NET Standard Stack from OPC Foundation requested some API changes in *OPC UA .NET Standard SDK*:
 - The *ClientSession.Factory* property type changed from *Opc.Ua.EncodeableFactory* to *Opc.Ua.IEncodeableFactory*.
 - The signature of *Initialize* method from *BaseComplexTypeValue* and all derived types changed to *Initialize(IEncodeableFactory factory)*.
 - The *IEncodeableFactory* extension methods from OPC UA .NET Standard SDK Client Library have a new parameter of type *Softing.Opc.Ua.Client.Types.Domain*. This parameter allows the usage of both Server and Client libraries in the same application without custom data type implementation interfering from one domain to the other.

The methods that have this new parameter are: *GetComplexTypeInfo()*, *GetTypeTable()*. Two helper methods were added to the *ClientSession*: *GetComplexTypeInfo()* one with type id parameter and one with and type name.

- The *ClientCertificateLifeTime* property was added to *ClientToolkitConfiguration* to allow creating the client certificate with a custom lifetime.
- Bug fixes:
 - The loading of custom data types was improved to avoid problems if a monitored item receives data events with instances of custom data types while the data type information is loading. While the

data type information is loading the data notifications contain only the bytes not the decoded instance of StructuredValue.

SDK libraries are compiled using **OPCFoundation.NetStandard.Opc.Ua - version 1.4.367.42.** . When using OPC UA .NET Standard SDK version 3.00 ***it is mandatory to use the same version of OPC UA .NET Standard Stack from OPC Foundation.***

■ Version 2.90

- The **PubSub Library** was removed from the OPC UA .NET Standard SDK. The *PubSub* implementation is now part of OPC UA .NET Standard Stack from OPC Foundation and the *Publisher* and *Subscriber* sample applications were modified accordingly.

Note: PubSub Library from OPC UA .NET Standard Stack from OPC Foundation is currently in beta version because there will be breaking changes in future release. For applications developed with OPC UA .NET Standard SDK version 2.90 it is mandatory to add reference to OPCFoundation.NetStandard.Opc.Ua.PubSub - version 1.4.366.38-beta NuGet package.

- The *Publisher* and *Subscriber* sample applications were modified to include handling for the implementation of MQTT with JSON and UADP encoding that was recently added to OPC UA .NET Standard Stack from OPC Foundation.

- OPC UA .NET Standard SDK Server Library:

- The custom structure data type was extended to support fields that have the *ValueRank* greater than *OneDimension*. SampleServer project provides sample code for creating a structured value with *TwoDimension* fields and also it creates a *Variable* node that uses this data type for its value.
- In *NodeManager* class:
 - The *NodeManager.CreateDataType()* method was modified to add also the *Default XML* encoding node for the newly created data type node.
Note: This change is possible to shift node ids if they are created using the default *New()* method.
 - The *NodeManager.GetDefaultValueForDatatype()* method can create values with *ValueRank* greater than *OneDimension*.
- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- The *ClientSession.GetDefaultValueForDatatype()* method can create values with *ValueRank* greater than *OneDimension*.
- Bug fixes.

SDK libraries are compiled using **OPCFoundation.NetStandard.Opc.Ua - version 1.4.366.38.** . When using OPC UA .NET Standard SDK version 2.90 **it is mandatory to use the same version of OPC UA .NET Standard Stack from OPC Foundation.**

☒ Version 2.80

- Integrates the new version of **OPC UA .NET Standard Stack from OPC Foundation** (*NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.365.48*) which includes:
 - The fixes for the following cybersecurity vulnerabilities: CVE: 2021-27432, CVE: 2020-29457.
 - *TranslateBrowsePathsToNodeIds* fix: In *RelativePathElement* objects the *IncludeSubtypes* should be ignored when the *ReferenceTypeId* is not specified.

- OPC UA .NET Standard SDK Server Library:

- In *NodeManager* class:
The existing *ReportEvent* method became obsolete and a new version of *ReportEvent* method was created with a different set of parameters.
- *SampleServer* project was modified to reference the fine grained *OPCFoundation.NetStandard.Opc.Ua.* NuGet* packages as described in [Reduce Server's Output Folder Size](#) section.
- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- *SampleClient* project was modified to reference the fine grained *OPCFoundation.NetStandard.Opc.Ua.* NuGet* packages as described in [Reduce Client's Output Folder Size](#) section.
- Bug fixes.

SDK libraries are compiled using **OPCFoundation.NetStandard.Opc.Ua - version 1.4.365.48.** When using OPC UA .NET Standard SDK version 2.80 **it is mandatory to use the same version of OPC UA .NET Standard Stack from OPC Foundation.**

☒ Version 2.70

- Support for VisualStudio 2019.

- OPC UA .NET Standard SDK Server Library:

- Include the [Amendment OPC 10001-13: OPC Unified Architecture](#).
- Import NodeSet2 XML sets the *Parent* node for nodes that have *ParentNodeId* property.
- The *XamarinSampleServer* target android version was upgraded to version 9.0 (Pie).
- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- Support for *Register* and *Unregister Nodes*.

- The *XamarinSampleClient* target android version was upgraded to version 9.0 (Pie).
- Bug fixes.

- OPC UA .NET Standard SDK PubSub Library:

- Bug fixes.

SDK libraries are compiled using OPCFoundation.NetStandard.Opc.Ua - version 1.4.364.40. When using OPC UA .NET Standard SDK version 2.70 it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 2.60

- OPC UA .NET Standard SDK Server Library:

- Support for reverse connectivity.
- In *NodeManager* class:
 - The *CreateObjectFromType()* and *CreateVariableFromType()* methods were enhanced with an additional parameter (*createOptionalProperties*) that specifies if the properties that have optional modelling rule are instantiated when creating an object/variable. Note that the types from OPC UA .NET Standard Stack from OPC Foundation will not create optional properties.
 - The *ReportEvent* method was enhanced to accept also the node id of the event type that will be reported.
- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- Support for reverse connectivity.
- Asynchronously methods added for Connect (*ConnectAsync*), Disconnect (*DisconnectAsync*), GetEndpoints (*GetEndpointsAsync*), DiscoverServers (*DiscoverServersAsync*) and DiscoverServersOnNetwork (*DiscoverServersOnNetworkAsync*).
- Bug fixes.

- OPC UA .NET Standard SDK PubSub Library:

- Bug fixes.

SDK libraries are compiled using the NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.363.49. When using OPC UA .NET Standard SDK version 2.60 it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 2.50

- OPC UA .NET Standard SDK Server Library:

- Internal support for AccessRestrictions, RolePermissions and UserRolePermissions attributes.

- Support for GDS and Push Management of application certificates and trust lists.
- Bug fixes.

- OPC UA .NET Standard SDK Client Library:

- Support for GDS and Pull Management of application certificates and trust lists.
- ReadNode method from *ClientSession* will not throw a *ServiceResultException* when the node is not found, but will return an instance of *VariableNodeEx*. Status codes for each attribute are available via *GetStatusCode()* method.
- Bug fixes.

- OPC UA .NET Standard SDK PubSub Library:

- Bug fixes.

- .NET Framework support updated to version **4.6.2** in accordance with *OPC UA .NET Standard Stack from OPC Foundation*.

- The support for **Visual Studio 2015** was removed.

SDK libraries are compiled using the NuGet package *OPCFoundation.NetStandard.Opc.Ua* - version 1.4.362.42. When using OPC UA .NET Standard SDK version 2.50 it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 2.40

- OPC UA .NET Standard SDK Server Library:

- Support for creating custom *DataType*, *VariableType* and *ObjectType* nodes and instances
- Extended support for complex data types:
 - Support for custom *Structure*, *StructureWithOptionalFields*, *Union*, *Enumeration* and *OptionSet* types.
 - Expose *DataTypeDefinition* attribute of *DataType* nodes to allow clients to read custom data types.
 - Enhanced *ImportNodeSet* functionality for reading the *DataTypeDefinition* attribute of *DataType* nodes.
 - Sample code for creating and handling custom types.
- Support for creating custom variable and object types.
- Bug fixes.
- .NET Standard 1.4 support removed.

- OPC UA .NET Standard SDK Client Library:

- Extended support for complex data types:
 - Support for custom *Structure*, *StructureWithOptionalFields*, *Union*, *Enumeration* and *OptionSet* types.
 - Load server custom data types by reading the *DataTypeDefinition* attribute of *DataType* nodes for V1.04 servers.
 - Old setting *DecodeCustomDataTypes* changed its semantic and it configures if the custom data types information is loaded from *DataTypeDefinition* attribute.

- New setting `DecodeDataTypeDictionaries` has the semantics of old `DecodeCustomDataTypes` setting and configures if the custom data types information is loaded from data type dictionaries.
- Sample code for handling custom data types.
- Bug fixes.
- .NET Standard 1.4 support removed.

- OPC UA .NET Standard SDK PubSub Library:

- Bug fixes.

SDK libraries are compiled using the NuGet package `OPCFoundation.NetStandard.Opc.Ua` - version 1.4.359.31. When using OPC UA .NET Standard SDK version 2.40 it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 2.30

- OPC UA .NET Standard SDK PubSub Library:

- Provides simplified API for configuring a PubSub application.
- `SampleServer` provides a new dedicated node manager that uses **OPC UA .NET Standard SDK PubSub Library** and behaves like a Publisher and Subscriber.
- Changes in licensing namespaces:

In version 2.20 the `PubSub` licensing classes were part of `Softing.Opc.Ua.Private` namespace. Starting with version 2.30 the `PubSub` licensing classes are part of `Softing.Opc.Ua.PubSub` namespace. Please change your code to point to the new namespace.

- OPC UA .NET Standard SDK Server Library:

- Bug fixes.
- Changes in licensing namespaces:

In version 2.20 the `Server` licensing classes were part of `Softing.Opc.Ua.Private` namespace. Starting with version 2.30 the `Server` licensing classes are part of `Softing.Opc.Ua.Server` namespace. Please change your code to point to the new namespace.

- OPC UA .NET Standard SDK Client Library:

- Bug fixes.
- Changes in licensing namespaces:

In version 2.20 the `Client` licensing classes were part of `Softing.Opc.Ua.Private` namespace. Starting with version 2.30 the `Client` licensing classes are part of `Softing.Opc.Ua.Client` namespace. Please change your code to point to the new namespace.

SDK libraries are compiled using the NuGet package `OPCFoundation.NetStandard.Opc.Ua` - version 1.4.357.28. When using OPC UA .NET Standard SDK version 2.30 it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 2.20

The initial version of **OPC UA .NET Standard SDK PubSub Library**:

- Provides an API for simplified OPC UA Publisher and Subscriber development using UDP transport and binary encoded messages.
- Provides SamplePublisher and SampleSubscriber projects that use the simplified API from **dataFEED OPC UA .NET Standard SDK PubSub Library**.

- **OPC UA .NET Standard SDK Server Library:**

- Enhanced ImportNodeSet functionality that allows instantiating data types loaded from dictionaries.
- New method *GetDefaultValueForDatatype* was added to *NodeManager*. It creates and returns the default value for the specified data type, including data types imported from NodeSet2 XML files.
- Added support for **Aes256_Sha256_RsaPss** security policy.

- **OPC UA .NET Standard SDK Client Library:**

- *ClientSession.KeepAlive* handling was moved on a separate thread to avoid possible deadlocks.

SDK libraries are compiled using the NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.355.26. When using OPC UA .NET Standard SDK version 2.20 it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 2.10

- "Softing OPC UA .NET Standard Toolkit" was renamed to "**OPC UA .NET Standard SDK**".

- OPC UA Server Compliance Certification.

- **OPC UA .NET Standard SDK Server Library:**

- Enhanced ImportNodeSet functionality that handles duplicate nodes.
- Added Export NodeSet functionality.
- Added create instance from OPC UA type specified by *NodeId* functionality.
- Added sample code for file transfer.
- Added sample code for temporary file transfer.

- Added support for **Aes128_Sha256_RsaOaep** security policy.

- Removed **Windows 8.1** support due to known security limitations incompatible with OPC UA .NET Standard SDK implementation.

- Properties marked as deprecated in *ServerToolkitConfiguration*.

SDK libraries are compiled using the NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.3.354.23. When using OPC UA .NET Standard SDK version 2.10 it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 2.00

Version 2.00 provides the following features for **Softing OPC UA .NET Standard Client Toolkit** (the product was renamed to "**OPC UA .NET Standard SDK Client Library**" in version 2.10):

- DLL name was changed from *Softing.Opc.Ua.dll* to *Softing.Opc.Ua.Client.dll*.

Namespace changes:

- All entities from namespace *Softing.Opc.Ua* were moved to namespace *Softing.Opc.Ua.Client* (*SecurityPolicy* enum, *SDK Utils* and *UaApplication* classes)
- All entities from namespace *Softing.Opc.Ua.Private* were moved to *Softing.Opc.Ua.Client.Private* (*License* class and *LicenseFeature* enum)
- All entities from namespace *Softing.Opc.Ua.Types* were moved to *Softing.Opc.Ua.Client.Types*.
- *ApplicationConfigurationEx* class available in version 1.00 and 1.10 in namespace *Softing.Opc.Ua* was removed. We are using the *ApplicationConfiguration* class available in OPC UA .NET Standard Stack from OPC Foundation . A new configuration class *ClientToolkitConfiguration* was added to the Softing OPC UA .NET Standard Client Toolkit, it is used as a configuration extension as described in Client configuration section.
- New method added to *ClientSubscription* class. *ConditionRefresh* method tells the server to refresh all conditions (alarms) being monitored by the subscription.

Version 2.00 provides the initial version of **Softing OPC UA .NET Standard Server Toolkit**:

- Provides an API for simplified server development. The OPC UA .NET Standard SDK Server Library offers a simplified approach for Data Access, Methods and Standard Event Subscriptions
- SampleServer and XamarinSampleServer are refactored to use the new Softing OPC UA .NET Standard Server Toolkit (*Softing.Opc.Ua.Server.dll*).

SDK libraries are compiled using the NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.353.15. When using OPC UA .NET Standard SDK version 2.00 it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 1.10

Version 1.10 provides the following features for **Softing OPC UA .NET Standard Client Toolkit**:

- Support for Linux distributions: Ubuntu 16.04 and Debian 9.30.
- Support for Android. Sample Client and Sample Server for Android implemented with Xamarin Forms were added in the samples folder.
- Softing OPC UA .NET Standard Client Toolkit is compiled using NuGet package *OPCFoundation.NetStandard.Opc.Ua* - version 1.3.352.10. When using version 1.10 of the Softing OPC UA .NET Standard Toolkit it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

☒ Version 1.00

Initial release of **Softing OPC UA .NET Standard Toolkit** provides only **Softing OPC UA .NET Standard Client Toolkit** functionality.

Note: The product was renamed to "**OPC UA .NET Standard SDK**" in version 2.10.

New license keys. The old license keys (valid for OPC UA .NET SDK 1.4x) are not valid for the new version anymore.

The known easy client API (familiar to OPC UA .NET SDK 1.4x users) ported to the .NET Standard Stack. Unfortunately it required refactoring of the API interface itself, therefore the backward compatibility (at code level) is broken. We addressed that by not making it too diverse compared to the previous one and by offering within this documentation a migration guide.

Windows support only. This is NOT meaning the Softing OPC UA .NET Standard Toolkit cannot compile and run on other platforms supported by .NET Standard; it means for the current version the tests were performed only on Windows OS.

Discontinued support for client side redundancy. It will be refactored and completed in the next versions.

For those familiar with the support of the complex data types decoding in the OPC UA .NET SDK 1.4x we offer a continuation of it as we observed that many integrators appreciated it.

Softing OPC UA .NET Standard Client Toolkit is compiled using the NuGet package *OPCFoundation.NetStandard.Opc.Ua* - version 1.3.350.4. When using version 1.00 of the Softing OPC UA .NET Standard Toolkit it is important to use the same version of OPC UA .NET Standard Stack from OPC Foundation.

1.3 System Requirements

The version 3.50 of the OPC UA .NET Standard SDK requires one of the following supported operating systems:

- Windows 7 SP1
- Windows 10
- Windows 11
- Windows Server 2012 SP1
- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2022
- Linux (see Note)
 - Tested under Debian 10 (no development support, only runtime)
 - Tested under Ubuntu 20.04 (no development support, only runtime)

Note: Although OPC UA .NET Standard SDK should run on all Microsoft supported Linux distributions which can host the .NET Core runtime or SDK, it has been thoroughly tested only on Debian 10 and Ubuntu 20.04.

The development package requires support from Microsoft .NET 6.0, Microsoft .NET 4.6.2, Microsoft .NET Standard 2.0 / 2.1 or .NET Core 3.1.

The installation deploys projects and solutions for Microsoft Visual Studio IDE, therefore at least one of the following versions should be installed:

IDE version	Prerequisites
Visual Studio 2022	.NET 6.0 SDK - https://dotnet.microsoft.com/en-us/download/dotnet/6.0
	.NET Core 3.1 SDK - https://dotnet.microsoft.com/download/dotnet-core/3.1
	.NET 4.6.2 SDK - https://dotnet.microsoft.com/download/dotnet-framework/net462
Visual Studio 2019	.NET Core 3.1 SDK - https://dotnet.microsoft.com/download/dotnet-core/3.1
	.NET 4.6.2 SDK - https://dotnet.microsoft.com/download/dotnet-framework/net462
Visual Studio 2017	.NET 4.6.2 SDK - https://dotnet.microsoft.com/download/dotnet-framework/net462

Note: We recommend the usage of Visual Studio 2022 but all the solutions provided by OPC UA .NET Standard SDK can be opened using Visual Studio 2019 (without support for .NET 6.0) or using Visual Studio 2017 but only for .NET 4.6.2.

1.4 OPC UA Supported Features

Communication protocols

- opc.tcp - binary encoding
- https - binary encoding

Note: The HTTPS protocol is supported and tested except interoperability fully testing.

It is available to be enabled in both Server and Client applications. The available HTTPS functionality is entirely implemented in OPC UA .NET Standard Stack from OPC Foundation.

Server

- Data Access Server
- Method Server
- DataChange Subscription Server
- Standard Event Subscription Server
- Address Space Notifier Server
- Standard UA Server
- Local Discovery Server
- Core Server
- History Data Access Server
- Alarms and Conditions Server

Client

- Event Subscriber Client
- DataAccess Client
- Method Client
- DataChange Subscriber Client
- AddressSpace Lookup Client
- Discovery Client
- Core Client
- History Data Access Client
- Alarms and Conditions Client

Unsupported Features

- Cancel Service
- Auditing Service Set
- TransferSubscriptions Service
- NodeManagement Service Set
 - AddNodes Service
 - AddReferences Service
 - DeleteNodes Service
 - DeleteReferences Service
- Query Service Set
 - QueryFirst Service
 - QueryNext Service

2 Client API Migration Guide

OPC UA .NET Standard SDK introduces breaking changes for previous versions of OPC UA SDK.

In **old SDK version** (<%OEMDATAFEEDDOTNET%>) most of the stack classes had a corresponding wrapped class with the same name. In **OPC UA .NET Standard SDK** we kept only those wrapped classes that added functionality and they have the stack name suffixed by *Ex* from *Extension* (e.g. *ReferenceDescriptionEx*). Old code that needs to be migrated must rename those classes that have been extended in **OPC UA .NET Standard SDK** to their new names and point to stack namespaces for those classes that do not have a wrapper any more.

Note: Each application that uses OPC UA .NET Standard SDK version 3.50 must add reference to NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96. The NuGet package contains the OPC UA stack implementation maintained by OPC Foundation under [GIT repository](#).

In **OPC UA .NET Standard SDK** we added *Client* prefix to *Session*, *Subscription*, *MonitoredItem* and *Method* classes to be able to easily differentiate between those and the ones provided by the stack. Therefore there are *ClientSession*, *ClientSubscription*, *ClientMonitoredItem* and *ClientMethod* classes.

General

1. Copy your solution folder to a new location.
2. Open copy solution in Visual Studio 2022.
3. Change target framework to .Net Framework 4.6.2. in project properties window (required by .net standard 2.0 libraries).
4. Remove reference to old *Softing.Opc.Ua.Toolkit.dll* from your project.
5. Add reference to *Softing.Opc.Ua.Client.dll* to your project.
6. Add reference to NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96. (Or add reference to the fine grained OPC Foundation NuGet packages as described in [Reduce Client's Output Folder Size](#) section).
7. Rename namespaces as described in table below - section 1.
8. Create application configuration. You can refactor code like in table below - section 2.1, or you can create a configuration file similar to the one provided with SampleClient application, rename it, change it for your needs and use it. **Please do not change the order of XML elements in configuration file.**
9. Apply **Application** refactor explained in table below - section 3.
10. Use *ClientSession* from *Softing.Opc.Ua.Client* instead of *Softing.Opc.Ua.Toolkit.Client.Session*. See table below - section 5.
11. Instantiate *ClientSession* with *uaApplication.CreateSession* like described in section 5.1 (table). If there is user authentication logic please use table - section 5.2 for refactor details.

Discovery functionality

1. Apply all that applies from **General**.
2. Apply refactor from table section 4 (4.1, 4.2).

Connect Functionality

1. Apply all that applies from **General**.
2. Apply refactor from table section 5 (5.1, 5.2).

Browse functionality

1. Apply all that applies from **General**.
2. Remove handling of *Session.ContinuationPointReached* event, see table below section **7.1**.
3. Use *BrowseDescriptionEx* instead of *BrowseOptions* (table - sections **7.1, 7.2**).
4. Use *ReferenceDescriptionEx* instead of *ReferenceDescription* for *Browse* methods return (table section **7.4**).
5. To transform *ExpandedNodeID* to *NodeID* for *Browse* methods use refactor from table - section **8**.

TranslateBrowsePath functionality

1. Apply all that applies from **General**.
2. Rename *BrowsePath* to *BrowsePathEx* (table section **7.5**).
3. Rename *BrowsePathResult* to *BrowsePathResultEx* (table section **7.6**).

Data Access

1. Apply all that applies from **General**.
2. Rename *Subscription* class to *ClientSubscription* (table section **6**).
3. Rename *MonitoredItem* to *ClientMonitoredItem* (table sections **9, 9.1, 10**).
4. Replace *AttributID* with *Attributes* (table section **10**).

Read complex values or enum values

1. Apply all that applies from **Data Access**.
2. Ensure that type dictionaries are loaded in current *ClientSession* and then use *DataValueEx.ProcessedValue* field to retrieve complex value/enum value/values. Complex values are instances of *StructuredValue*, *OptionalFieldsStructuredValue*, *UnionStructuredValue* and enum types are instances of *EnumValue*. They are defined in Softing.Opc.Ua.Client.dll as opposed to old SDK version where *StructuredValue* and *EnumValue* types were part of the stack.

Table section **12.1** describes the approach.

Write complex values

1. Apply all that applies from **Data Access**.
2. Ensure that type dictionaries are loaded in current *ClientSession*
3. Check sample code *ReadWriteClient.WriteComplexValue()* for a concrete example of how to write a complex value. The example first reads the type information for the node id, then gets default value for that type and then changes the default value to desired value and writes it to the node. See table **section 12.2**.

Monitored item

1. Apply all that applies from **Data Access**.
2. Use *EventFilterEx* instead of stack *EventFilter* and *SelectOperandEx* instead of stack *SelectOperand*.

Handling alarms

1. Apply all that applies from **Data Access**.
2. Use *EventFilterEx* instead of *EventFilter* to specify the event filter for alarms. See table section **9.2**.

History read

1. Apply all that applies from **General**.
2. Remove *HistoryContinuationPointReached* event handling from session. *ClientSession* implementation does not provide this event any more. (see table section **5.3**).
3. **Read raw** changes are described in table section **13.1**.

4. **Read at time** changes are described in table section **13.2**.
5. **Read processed** changes are described in table section **13.3**.

Method call - see table section **14**.

Logging - see table section **15**.

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
1	Namespaces	
1.1	Softing.Opc.Ua.Toolkit	remove
1.2	Softing.Opc.Ua.Toolkit.Client	Softing.Opc.Ua.Client
1.3	Softing.Opc.Ua.Toolkit.Client.Nodes	Softing.Opc.Ua.Client.Nodes
1.4	Softing.Opc.Ua.Sdk.* (old Softing OPC Ua SDK)	Opc.Ua.*
2	Initialize configuration	
2.1	<pre> Application.Configuration.ApplicationName = "SoftingOpcUaBrowseClient"; Application.Configuration.ProductUri = "http://industrial.softing.com/OpcUaNetToolkit/B rowseClient"; // security configuration. string applicationFolder = Path.Combine(Assembly.GetExecutingAssembly().Loc ation, @"..\..\..\..\..\..\.."); applicationFolder = Path.GetFullPath(applicationFolder); Application.Configuration.Security.ApplicationCe rtificateStore = Path.Combine(applicationFolder, @"c:\tmp\pki\own"); Application.Configuration.Security.ApplicationCe rtificateSubject = Application.Configuration.ApplicationName; Application.Configuration.Security.TrustedCertif icateStore = Path.Combine(applicationFolder, @"c:\tmp\pki\trusted"); Application.Configuration.Security.TrustedIssuer CertificateStore = Path.Combine(applicationFolder, @"c: \temp\pki\issuer"); Application.Configuration.Security.RejectedCertifi cateStore = Path.Combine(applicationFolder, @"c:\tmp\pki\rejected"); Application.CertificateValidation += Application_CertificateValidation; </pre>	<pre> // Create the ApplicationConfiguration object ApplicationConfiguration configuration = new ApplicationConfiguration(); configuration.ApplicationName = "SoftingOpcUaBrowseClient"; configuration.ProductUri = "http://industrial.softing.com/OpcUaNetToolkit/B rowseClient"; // Create new instance of ClientToolkitConfiguration and set desired properties ClientToolkitConfiguration clientTkCfg = new ClientToolkitConfiguration(); clientTkCfg.DiscoveryOperationTimeout = 10000; // Set ClientToolkitConfiguration instance as an extension of current ApplicationConfiguration object configuration.UpdateExtension<ClientToolkitConfig uration>(new XmlQualifiedName("ClientToolkitConfiguration"), clientTkCfg); // security configuration. string applicationFolder = Path.Combine(Assembly.GetExecutingAssembly().Loc ation, @"..\..\..\..\..\..\.."); applicationFolder = Path.GetFullPath(applicationFolder); configuration.SecurityConfiguration = new SecurityConfiguration { </pre>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
	<pre>// The Validate() method ensures that the specified configuration is valid. try { Application.Configuration.Validate(); } catch (Exception ex) { Console.WriteLine(ex.Message); return false; } // trace configuration Application.Configuration.Trace.LogFileName = @"Logs\BrowseClient.txt"; Application.Configuration.Trace.LogFileMaxSize = 10; Application.Configuration.Trace.LogFileMaxRollBackups = 5; Application.Configuration.Trace.LogFileTracelevel = TraceLevels.Warning; //enable all masks Application.Configuration.Trace.LogFileTraceMask = 0x00FF00FF; Application.Configuration.Trace.Tracelevel = TraceLevels.Warning; //enable all masks Application.Configuration.Trace.TraceMask = 0x00FF00FF;</pre>	<pre>ApplicationCertificate = new CertificateIdentifier { SubjectName = configuration.ApplicationName, StoreType = CertificateStoreType.Directory, StorePath = @"c:\tmp\pki\own" }, TrustedPeerCertificates = new CertificateTrustList { StoreType = CertificateStoreType.Directory, StorePath = @"c:\tmp\pki\trusted", }, TrustedIssuerCertificates = new CertificateTrustList { StoreType = CertificateStoreType.Directory, StorePath = @"c:\tmp\pki\issuer", }, RejectedCertificateStore = new CertificateTrustList { StoreType = CertificateStoreType.Directory, StorePath = @"c:\tmp\pki\rejected", }, AutoAcceptUntrustedCertificates = true }; // trace configuration configuration.TraceConfiguration = new TraceConfiguration() { OutputFilePath = @"Logs\BrowseClient.txt", DeleteOnLoad = true, TraceMasks = 519 };</pre> <p>Code snippet is from ConnectClient sample.</p>
3	Application	
3.1	<p>Application class was used to instantiate and configure an OPC Ua client application.</p> <pre>Application.Configuration.ApplicationName = "UA Sample Client";</pre>	<p>UaApplication class replaces Application class. UaApplication is not static class, the user can instantiate and configure multiple UaApplications as opposed to old Application approach.</p> <pre>// Create the UaApplication object from config file</pre>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
	<pre>Application.Configuration.ProductUri = "http://industrial.softing.com/OpcUaNetToolkit/BrowseClient"; //configure the application by code ...</pre>	<pre>UaApplication application = UaApplication.Create(Constants.ConfigurationFile) .Result;</pre> <p><i>Code snippet is from Program.cs in sample project.</i></p> <p>Or</p> <pre>// Create the ApplicationConfiguration object ApplicationConfiguration configuration = new ApplicationConfiguration(); configuration.ApplicationName = "UA Sample Client"; //configure the application by code ... // Create the UaApplication object from config object UaApplication application = UaApplication.Create(configuration).Result;</pre> <p><i>Code snippet is from ConnectClient sample.</i></p>
3.2	<p>Application.CertificateValidation event</p> <pre>Application.CertificateValidation += Application_CertificateValidation; private static void Application_CertificateValidation(object sender, CertificateValidationEventArgs e) { // Add custom logic for validating the server certificate. // Accept this certificate during the runtime of the application. e.ValidationOption = CertificateValidationOption.AcceptOnce; }</pre>	<p>CertificateValidation event was not created into UaApplication class. You must use CertificateValidator.CertificateValidation event instead.</p> <pre>application.Configuration.CertificateValidator.Ce rtificateValidation += Application_CertificateValidation; private static void Application_CertificateValidation(object sender, CertificateValidationEventArgs e) { // Add custom logic for validating the server certificate. // Accept this certificate during the runtime of the application. e.Accept = true; } <p>Note: If the CertificationValidation event is handled and one certificate is accepted the certificate is not copied into the <i>trusted</i> folder like it was while using <% OEMDATAFEEDDOTNET%>. The following code shows how to copy the trusted certificate in the <i>trusted</i> folder:</p> <pre>private void CertificateValidator_CertificateValidation(Certif icateValidator sender, CertificateValidationEventArgs e)</pre> </pre>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
		<pre> { try { ICertificateStore certificateStore = m_application.Configuration.SecurityConfiguration .TrustedPeerCertificates.OpenStore(); if (certificateStore != null) { // check for certificate file. var entry = certificateStore.FindByThumbprint(e.Certificate.T humbprint).Result; if (entry == null entry.Count == 0) { certificateStore.Add(e.Certificat e).Wait(); } } catch (Exception ex) { // log exception } } } </pre>
4	<i>Discovery</i>	
4.1	<p><i>Application.DiscoverServers(discoveryUrl)</i> method used to return a list of <i>EndpointDescription</i> objects for all servers found at <i>discoveryUrl</i>.</p> <pre> // The method will return all the available server endpoints from the specified machine IList<EndpointDescription> endpoints = Application.DiscoverServers(discoveryUrl); </pre> <p>To get the endpoints you need to call static <i>Application.GetEndpoints(serverDiscoveryUrl)</i> returns a list of <i>EndpointDescription</i> objects, one for each endpoint found on address.</p> <pre> // retrieve available endpoints for specified serverDiscoveryUrl. string serverDiscoveryUrl = applicationDescription.DiscoveryUrls[0]; IList<EndpointDescriptionEx> endpoints = uaApplication.GetEndpoints(serverDiscoveryUrl); </pre>	<p><i>UaApplication.DiscoverServers(discoveryUrl)</i> returns a list of <i>ApplicationDescription</i> objects, one for each server found on address.</p> <pre> // the method will return all the registered server applications from the specified machine. IList<ApplicationDescription> appDescriptions = uaApplication.DiscoverServers(discoveryUrl); </pre> <p>To get the endpoints you need to call <i>UaApplication.GetEndpoints(serverDiscoveryUrl)</i> returns a list of <i>EndpointDescriptionEx</i> objects, one for each endpoint found on address.</p> <pre> // retrieve available endpoints for specified serverDiscoveryUrl. string serverDiscoveryUrl = applicationDescription.DiscoveryUrls[0]; IList<EndpointDescriptionEx> endpoints = uaApplication.GetEndpoints(serverDiscoveryUrl); </pre> <p>Code snippets are from <i>DiscoveryClient</i> sample.</p>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
4.2	EndpointDescription	EndpointDescriptionEx
5	Session	ClientSession
5.1	<p>Session instances are created using constructor.</p> <pre>// Create the Session object. Session session = new Session(serverUrl, securityMode, securityPolicy.ToString(), messageEncoding, userId, null);</pre>	<p>There is no public constructor for <i>ClientSession</i>. Sessions are created by <i>UaApplication</i> class with <i>CreateSession</i> methods.</p> <p><i>Code snippet from ConnectClient sample:</i></p> <pre>// Create the Session object. ClientSession session = uaApplication.CreateSession(serverUrl, securityMode, securityPolicy, messageEncoding, userId, null);</pre> <p><i>Code snippet from AlarmsClient sample:</i></p> <pre>// create the session object with no security and anonymous login ClientSession session = uaApplication.CreateSession(Constants.ServerUrl);</pre>
5.2	<p>AnonymousUserIdentity</p> <pre>UserIdentity userIdentity = new AnonymousUserIdentity();</pre> <p>UserNameUserIdentity</p> <pre>UserIdentity userIdentity = new UserNameUserIdentity("usr", "pwd");</pre> <p>CertificateUserIdentity</p> <pre>UserIdentity userIdentity = CertificateUserIdentity(certificateFilePath, password);</pre>	<p>AnonymousUserIdentity class was removed. To be replaced with UserIdentity</p> <pre>UserIdentity userIdentity = new UserIdentity();</pre> <p>UserNameUserIdentity class was removed. To be replaced with UserIdentity</p> <pre>UserIdentity userIdentity = new UserIdentity("usr", "pwd");</pre> <p>CertificateUserIdentity class was removed. To be replaced with UserIdentity</p> <pre>X509Certificate2 certificate = new X509Certificate2(certificateFilePath, null as string, X509KeyStorageFlags.MachineKeySet X509KeyStorageFlags.Exportable); if (certificate != null) { // create UserIdentity from certificate UserIdentity certificateUserIdentity = new UserIdentity(certificate); }</pre> <p><i>Code snippet from ConnectClient sample.</i></p>
5.3	Session.HistoryContinuationPointReached	ClientSession does not provide a HistoryContinuationPointReached event.
5.4	RedundantSession	Not Available
6	Subscription	ClientSubscription

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
6.1	RedundantSubscription	Not Available
7	Browse	
7.1	<p>Session has an event ContinuationPointReached for controlling browse.</p> <pre data-bbox="148 481 825 572"><code>IList<ReferenceDescription> rootReferenceDescriptions = session.Browse(null, sender);</code></pre> <p>Browse with options:</p> <pre data-bbox="148 629 825 783"><code>BrowseOptions options = new BrowseOptions(); options.MaxReferencesReturned = 3; IList<ReferenceDescription> rootReferenceDescriptions = session.Browse(null, options, sender);</code></pre>	<p>ClientSession does not provide the ContinuationPointReached event for controlling browse.</p> <pre data-bbox="850 481 1543 572"><code>IList<ReferenceDescriptionEx> rootReferenceDescriptions = m_session.Browse(null);</code></pre> <p>Browse with options:</p> <pre data-bbox="850 629 1543 825"><code>BrowseDescriptionEx options = new BrowseDescriptionEx(); options.MaxReferencesReturned = 3; IList<ReferenceDescriptionEx> rootReferenceDescriptions = m_session.Browse(null, options);</code></pre> <p><i>Code snippet from BrowseClient sample.</i></p>
7.2	<p>BrowseOptions</p> <pre data-bbox="148 946 776 994"><code>BrowseOptions options = new BrowseOptions(); options.MaxReferencesReturned = 3;</code></pre>	<p>BrowseDescriptionEx</p> <pre data-bbox="850 946 1543 1036"><code>BrowseDescriptionEx options = new BrowseDescriptionEx(); options.MaxReferencesReturned = 3;</code></pre> <p><i>Code snippet from BrowseClient sample.</i></p>
7.3	<p>Parameter <i>cookie</i> from <i>Browse</i> methods.</p> <pre data-bbox="148 1178 776 1322"><code>public virtual IList<ReferenceDescription> Browse(NodeId nodeId, object cookie); public virtual IList<ReferenceDescriptionEx> Browse(NodeId nodeId, BrowseDescriptionEx browseOptions, object cookie);</code></pre>	<p>Parameter <i>cookie</i> was removed from <i>Browse</i> methods.</p> <pre data-bbox="850 1178 1543 1322"><code>public virtual IList<ReferenceDescriptionEx> Browse(NodeId nodeId); public virtual IList<ReferenceDescriptionEx> Browse(NodeId nodeId, BrowseDescriptionEx browseDescription);</code></pre>
7.4	ReferenceDescription	ReferenceDescriptionEx
7.5	BrowsePath	BrowsePathEx
7.6	BrowsePathResult	BrowsePathResultEx
8	<p>ExpandedNodeId to NodeId - wrapped classes from old SDK are used.</p> <pre data-bbox="148 1586 711 1676"><code>// transform ExpandedNodeId to NodeId NodeId nodeId = new NodeId(rootReferenceDescription.NodeId.Identifier , rootReferenceDescription.NodeId.NamespaceIndex);</code></pre>	<p>ExpandedNodeId to NodeId - stack classes are used.</p> <pre data-bbox="850 1543 1543 1719"><code>// transform ExpandedNodeId to NodeId NodeId nodeId = new NodeId(rootReferenceDescription.NodeId.Identifier , rootReferenceDescription.NodeId.NamespaceIndex);</code></pre> <p><i>Code snippet from BrowseClient sample.</i></p>
9	MonitoredItem	<p>ClientMonitoredItem</p> <p>The order of parameters in one of the constructors changed. Also, some parameters became optional.</p>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
	<pre>monitoredItem = new MonitoredItem(subscription, nodeId, AttributeId.Value, null, "Sample Monitored Item");</pre>	<pre>monitoredItem = new ClientMonitoredItem(subscription, nodeId, "Sample Monitored Item", Attributes.Value, null); // the code from above is similar to the next one since the default values for parameters are used monitoredItem = new ClientMonitoredItem(subscription, nodeId, "Sample Monitored Item");</pre>
9.1	<p>DataValue <i>Session.Read</i> returns <i>DataValue</i> <i>MonitoredItem.Read</i> returns <i>DataValue</i></p>	<p>DataValueEx <i>ClientSession.Read</i> returns <i>DataValueEx</i> <i>ClientMonitoredItem.Read</i> returns <i>DataValueEx</i> <i>DataValueEx.TryConvertToEnumValue</i> method - parameter order was changed, <i>session</i> parameter was moved on the first position.</p> <p>When writing a value using <i>ClientMonitoredItem.Write</i> or <i>ClientSession.Write</i> the <i>DataValue</i> object used for write</p>
9.2	<p>Create monitored item for alarms using EventFilter</p> <pre>// Configure the event filter EventFilter filter = filter = new EventFilter(); // specify the required fields of the events filter.AddSelectClause(ObjectTypes.BaseEventType, String.Empty, Attributes.NodeId); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.EventId); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.EventType); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.SourceNode); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.SourceName); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.Time); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.Message); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.Severity); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.EnabledState); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.ActiveState); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.AckedState); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.Comment); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.Retain);</pre>	<p>Create monitored item for alarms using EventFilterEx</p> <pre>// Configure the event filter EventFilterEx filter = filter = new EventFilterEx(); // specify the required fields of the events filter.AddSelectClause(ObjectTypes.BaseEventType, String.Empty, Attributes.NodeId); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.EventId); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.EventType); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.SourceNode); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.SourceName); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.Time); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.Message); filter.AddSelectClause(ObjectTypes.BaseEventType, BrowseNames.Severity); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.EnabledState); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.ActiveState); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.AckedState); filter.AddSelectClause(ObjectTypeIds.AcknowledgeableConditionType, BrowseNames.Comment);</pre>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
	<pre>// filter only for condition related events // (e.g in order to avoid audit events) filter.WhereClause.Push(FilterOperator.OfType, ObjectIds.ConditionType); // Create the MonitoredItem used to receive event notifications m_alarmsMonitoredItem = new MonitoredItem(m_subscription, m_alarmsModuleNodeId, "Alarms module", filter); m_alarmsMonitoredItem.EventsReceived += AlarmsMonitoredItem_EventsReceived;</pre>	<pre>filter.AddSelectClause(ObjectIds.AcknowledgeableConditionType, BrowseNames.Retain); // filter only for condition related events (e.g // in order to avoid audit events) filter.WhereClause.Push(FilterOperator.OfType, ObjectIds.ConditionType); // Create the MonitoredItem used to receive event notifications m_alarmsMonitoredItem = new ClientMonitoredItem(m_subscription, m_alarmsModuleNodeId, "Alarms module", filter); m_alarmsMonitoredItem.EventsReceived += AlarmsMonitoredItem_EventsReceived;</pre>
10	AttributeId	AttributeId enumeration was removed. Use constants with matching names from Attributes class.
11	DataTypeNode MethodNode ObjectNode ObjectTypeNode ReferenceTypeNode VariableNode VariableTypeNode ViewNode	DataTypeNodeEx MethodNodeEx ObjectNodeEx ObjectTypeNodeEx ReferenceTypeNodeEx VariableNodeEx VariableTypeNodeEx ViewNodeEx
12	Complex types/enum values	
12.1	Read complex value/enum value Enable custom types decoding and type dictionaries reload by setting configuration value before creating the session: Application.Configuration.Client.DecodeCustomDataTypes = true ; For enumeration values you need to call <i>TryConvertToEnumValue</i> on <i>DataValue</i> returned by <i>session.Read</i> . DataValue.Value is a <i>StructuredValue</i> or <i>EnumValue</i> instance if the complex value was read and it is set to something different from <i>null</i> .	Read complex type/enum value Enable custom types decoding and type dictionaries reload by setting configuration value before creating the <i>ClientSession</i> (by code or in XML configuration file): UaApplication.ClientToolkitConfiguration.DecodeCustomDataTypes = true ; UaApplication.ClientToolkitConfiguration.DecodeDataTypeDictionaries = true ; // trigger loading by calling session.FetchDataDictionaries(); // types are done loading when session.TypeDictionariesLoaded and/or DataTypeDefinitionsLoaded flags returns true For enumeration values you need to call <i>TryConvertToEnumValue</i> on <i>DataValueEx</i> returned by <i>session.Read</i> . The order of parameters for <i>TryConvertToEnumValue</i> changed. DataValueEx.ProcessedValue is an instance of <i>BaseComplexTypeValue</i> if a complex type value was read.

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
12.2	Write complex type <pre>//Browse path: Root\Objects\Refrigerators\Refrigerator #1\RefrigeratorStatus const string StaticValueNodeID = "ns=10;i=13"; WriteValue writeValue = new WriteValue(); writeValue.AttributeId = AttributeId.Value; writeValue.NodeId = new NodeId(StaticValueNodeID); DataValue valueToWrite = new DataValue(); ExpandedNodeId binaryEncodingId = new ExpandedNodeId ("nsu=http://industrial.softing.com/UA/Refrigerator;ns=0;i=444"); ExpandedNodeId typeId = new ExpandedNodeId ("nsu=http://industrial.softing.com/UA/Refrigerator;ns=0;i=435"); ExpandedNodeId xmlEncodingId = new ExpandedNodeId ("nsu=http://industrial.softing.com/UA/Refrigerator;ns=0;i=437"); XmlQualifiedName xmlQualifiedName = new XmlQualifiedName("RefrigeratorStatusType", typeId.NamespaceUri); StructuredValue complexData = StructuredValue.CreateNew(xmlQualifiedName, typeId, binaryEncodingId, xmlEncodingId); complexData.AddField("CondensorMotorRunning", true, new XmlQualifiedName("Boolean", "http://opcfoundation.org/BinarySchema/")); complexData.AddField("PressureBeforePump", (double) randomGenerator.Next(1, 110), new XmlQualifiedName("Double", "http://opcfoundation.org/BinarySchema/")); complexData.AddField("PressureAfterPump", (double) randomGenerator.Next(1, 70), new XmlQualifiedName("Double", "http://opcfoundation.org/BinarySchema/")); valueToWrite.Value = complexData; valueToWrite.ValueRank = ValueRanks.OneOrMoreDimensions; writeValue.Value = valueToWrite; try { StatusCode statusCode = session.Write(writeValue); }</pre>	Write complex type <pre>//Browse path: Root\Objects\Refrigerators\Refrigerator #1\RefrigeratorStatus const string StaticValueNodeID = "ns=10;i=13"; //read data type id for node StaticComplexNodeId ReadValueId readValueId = new ReadValueId(); readValueId.NodeId = new NodeId(StaticValueNodeID); readValueId.AttributeId = Attributes.DataType; Console.WriteLine("\n Read DataType Id for NodeId:{0}", StaticValueNodeID); DataValueEx dataValuetypeId = session.Read(readValueId); //Get Default value for data type StructuredValue defaultValue = session.GetDefaultValueForDatatype(dataValuetypeId.Value as NodeId, ValueRanks.Scalar) as StructuredValue; WriteValue writeValue = new WriteValue(); writeValue.AttributeId = Attributes.Value; writeValue.NodeId = new NodeId(StaticValueNodeID); DataValue valueToWrite = new DataValue(); valueToWrite.Value = defaultValue; writeValue.Value = valueToWrite; try { StatusCode statusCode = session.Write(writeValue); Console.WriteLine("\n The NodeId:{0} was written with the complex value {1} ", StaticValueNodeID, defaultValue.ToString()); Console.WriteLine(" Status code is {0}", statusCode); } catch (Exception e) { Console.WriteLine(e.Message); }</pre> <p>This sample is simplified, you do not need to know the types and their namespaces.</p> <p><i>StructuredValue.AddField method was removed.</i></p>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
	<pre data-bbox="148 297 825 587"> Console.WriteLine("\n The NodeId:{0} was written with the complex value {1} ", StaticValueNodeID, writeValue.Value.ToString()); Console.WriteLine(" Status code is {0}", statusCode); } catch (Exception e) { Console.WriteLine(e.Message); } </pre>	<p>The <code>BaseComplexTypeValue</code> type is the base type for all complex data types. The <code>StructuredValue</code> and <code>EnumValue</code> types extend it. There are new complex data types defined: <code>UnionStructuredValue</code>, <code>OptionalFieldsStructuredValue</code> and <code>OptionSetValue</code>.</p>
13	History read	
13.1	Read raw	<p>ReadRawArgument class was replaced with stack class ReadRawModifiedDetails.</p>
	<pre data-bbox="148 726 825 1115"> DateTime startTime = new DateTime(2011, 1, 1, 12, 0, 0); DateTime endTime = new DateTime(2011, 1, 1, 12, 1, 40); uint numberofValuesPerNode = 3; bool returnBounds = false; TimestampsToReturn timestampsToReturn = TimestampsToReturn.Both; ReadRawArgument argument = new ReadRawArgument(startTime, endTime, numberofValuesPerNode, returnBounds, timestampsToReturn); </pre> <p>Session.HistoryReadRaw signature changed.</p> <pre data-bbox="148 1402 825 1486"> public virtual List<DataValue> HistoryReadRaw(NodeId nodeToReadId, ReadRawArgument readRawArgument, object cookie); </pre>	<pre data-bbox="850 747 1543 1262"> DateTime startTime = new DateTime(2011, 1, 1, 12, 0, 0); DateTime endTime = new DateTime(2011, 1, 1, 12, 1, 40); uint numberofValuesPerNode = 3; bool returnBounds = false; TimestampsToReturn timestampsToReturn = TimestampsToReturn.Both; ReadRawModifiedDetails argument = new ReadRawModifiedDetails() { StartTime = startTime, EndTime = endTime, NumValuesPerNode = numberofValuesPerNode, ReturnBounds = returnBounds }; </pre> <p>ClientSession.HistoryReadRaw signature changed. The cookie parameter was removed because there is no <code>HistoryContinuationPointReached</code> event to match,</p> <pre data-bbox="850 1423 1543 1550"> public virtual List<DataValueEx> HistoryReadRaw(NodeId nodeToReadId, ReadRawModifiedDetails readRawModifiedDetails, TimestampsToReturn timestampsToReturn); </pre>
13.2	Read at time	<p>ReadAtTimeArgument class was replaced with stack class ReadAtTimeDetails.</p>
	<pre data-bbox="148 1634 825 1860"> List<DateTime> requiredTimes = new List<DateTime>(); requiredTimes.Add(new DateTime(2011, 1, 1, 12, 0, 0)); requiredTimes.Add(new DateTime(2011, 7, 1, 12, 1, 0)); bool useSimpleBounds = true; </pre>	<pre data-bbox="850 1655 1543 1839"> DateTimeCollection requiredTimes = new DateTimeCollection(); requiredTimes.Add(new DateTime(2011, 1, 1, 12, 0, 0)); requiredTimes.Add(new DateTime(2011, 7, 1, 12, 1, 0)); </pre>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
	<pre> TimestampsToReturn timestampsToReturn = TimestampsToReturn.Both; ReadAtTimeArgument argument = new ReadAtTimeArgument(requiredTimes, useSimpleBounds, timestampsToReturn); </pre> <p>Session.HistoryReadAtTime signature changed.</p> <pre> public virtual List<DataValue> HistoryReadAtTime(NodeId nodeToReadId, ReadAtTimeArgument readAtTimeArgument, object cookie); </pre>	<pre> ReadAtTimeDetails argument = new ReadAtTimeDetails() { ReqTimes = requiredTimes, UseSimpleBounds = true }; TimestampsToReturn timestampsToReturn = TimestampsToReturn.Both; </pre> <p>ClientSession.HistoryReadAtTime signature changed. The cookie parameter was removed because there is no HistoryContinuationPointReached event to match,</p> <pre> public virtual List<DataValueEx> HistoryReadAtTime(NodeId nodeToReadId, ReadAtTimeDetails readAtTimeDetails, TimestampsToReturn timestampsToReturn); </pre>
13.3	<p>Read processed</p> <pre> DateTime startTime = new DateTime (2011, 1, 1, 12, 0, 0); DateTime endTime = new DateTime (2011, 1, 1, 12, 1, 40); double processingInterval = 10000; List<NodeId> aggregateTypes = new List<NodeId>(); aggregateTypes.Add(new NodeId(2342)); //aggregate function average TimestampsToReturn timestampsToReturn = TimestampsToReturn.Both; ReadProcessedArgument argument = new ReadProcessedArgument(startTime, endTime, processingInterval, aggregateTypes, timestampsToReturn); </pre> <p>Session.HistoryReadAtTime signature changed.</p> <pre> public virtual List<DataValue> HistoryReadProcessed(NodeId nodeToReadId, ReadProcessedArgument readProcessedArgument, object cookie); </pre>	<p>ReadProcessedArgument class was replaced with stack class ReadProcessedDetails.</p> <pre> NodeIdCollection aggregateTypes = new NodeIdCollection(); aggregateTypes.Add(ObjectIds.AggregateFunction_Average); //aggregate function average </pre> <pre> ReadProcessedDetails argument = new ReadProcessedDetails() { StartTime = new DateTime(2011, 1, 1, 12, 0, 0), EndTime = new DateTime(2011, 1, 1, 12, 1, 40), ProcessingInterval = 10000, AggregateType = aggregateTypes }; TimestampsToReturn timestampsToReturn = TimestampsToReturn.Both; </pre> <p>ClientSession.HistoryReadProcessed signature changed. The cookie parameter was removed because there is no HistoryContinuationPointReached event to match,</p> <pre> public virtual List<DataValueEx> HistoryReadProcessed(NodeId nodeToReadId, ReadProcessedDetails readProcessedDetails, TimestampsToReturn timestampsToReturn); </pre>
14	<p>Async Method calls - CallCompleted event definition</p>	<p>Async Method calls - CallCompleted event definition changed because <i>MethodExecutionArgs</i> name changed to <i>MethodExecutionEventArgs</i>.</p>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
	<pre data-bbox="148 297 687 382">public event EventHandler<MethodExecutionEventArgs> CallCompleted;</pre> <p>MethodExecutionEventArgs</p> <pre data-bbox="148 508 719 656">private void Session_CallCompleted(object sender, MethodExecutionEventArgs e) { ... }</pre>	<pre data-bbox="842 297 1380 382">public event EventHandler<MethodExecutionEventArgs> CallCompleted;</pre> <p>MethodExecutionEventArgs - does not have Ex suffix it is not a wrapper class. Name changed according to EventArgs classes naming rules. <i>CallCompleted</i> event handler definition will change to:</p> <pre data-bbox="842 572 1543 720">private void Session_CallCompleted(object sender, MethodExecutionEventArgs e) { ... }</pre>
15	<p>Logging The old SDK uses the following log messages from <i>Softing.Opc.Ua.Sdk.Trace</i> class.</p> <pre data-bbox="148 840 784 1284">public void Log(TraceLevels traceLevel, TraceMasks traceMask, string objectId, string message); public void Log(TraceLevels traceLevel, TraceMasks traceMask, string objectId, string message, Exception exception); public void Log(TraceLevels traceLevel, TraceMasks traceMask, string objectId, string message, Exception exception, IMaskFormatProvider provider); public void Log(TraceLevels traceLevel, TraceMasks traceMask, string objectId, string message, Exception exception, IMaskFormatProvider provider, bool bypassTraceEventHandler);</pre> <p>Logging configuration:.</p> <pre data-bbox="148 1353 817 1860"><TraceConfiguration> <!-- Enable ALL --> <TraceMasks>0x00FF00FF</TraceMasks> <Log4NetConfiguration> <log4net> <appender name="RollingLogFileAppender" type="log4net.Appender.RollingFileAppender"> <file value="Logs\AlarmsClient.txt" /> <appendToFile value="true" /> <maxSizeRollBackups value="5" /> <maximumFileSize value="10MB" /> <rollingStyle value="Size" /> <staticLogFileName value="true" /> <layout type="log4net.Layout.PatternLayout"> <header value="[Header]\n;" /> <footer value="Footer\n;" /> </layout> </appender> </log4net> </Log4NetConfiguration> </TraceConfiguration></pre>	<p>Logging OPC UA .NET Standard SDK uses the following log messages from stack class <i>Softing.Opc.Utils</i> class.</p> <pre data-bbox="842 840 1511 1136">public static void Trace(int traceMask, string format, bool handled, params object[] args); public static void Trace(int traceMask, string format, params object[] args); public static void Trace(Exception e, string format, bool handled, params object[] args); public static void Trace(Exception e, string format, params object[] args); public static void Trace(string format, params object[] args);</pre> <p>Logging configuration:</p> <pre data-bbox="842 1290 1543 1902"><OutputFilePath>%CommonApplicationData% \Softing\OpcUaNetStandardToolkit\logs\SampleClient.log</OutputFilePath> <DeleteOnLoad>true</DeleteOnLoad> <!-- Show Only Errors --> <!-- <TraceMasks>1</TraceMasks> --> <!-- Show Only Security and Errors --> <!-- <TraceMasks>513</TraceMasks> --> <!-- Show Only Security, Errors and Trace --> <!-- <TraceMasks>515</TraceMasks> --> <!-- Show Only Security, COM Calls, Errors and Trace --> <!-- <TraceMasks>771</TraceMasks> --> <!-- Show Only Security, Service Calls, Errors and Trace --> <!-- <TraceMasks>523</TraceMasks> --> <!-- Show Only Security, ServiceResultExceptions, Errors and Trace --> <TraceMasks>519</TraceMasks> </TraceConfiguration></pre>

Id.	dataFEED .NET SDK 1.43	OPC UA .NET Standard SDK 3.50
	<pre data-bbox="143 291 763 646"><conversionPattern value="%date % thread %-5level - %message%newline" /> </layout> </appender> <root> <level value="WARN" /> <appender-ref ref="RollingLogFileAppender" /> </root> </log4net> </Log4NetConfiguration> </TraceConfiguration></pre>	

3 Reduce Client's Output Folder Size

The OPC UA .NET Standard SDK is build over the OPC UA .NET Standard Stack from OPC Foundation. The OPC UA .NET Standard Stack from OPC Foundation shall be referenced as NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96 in all *Client* applications. This NuGet reference will bring some unnecessary dlls in the output folder.

Starting with version **2.80** it is possible to trim down the number of assemblies copied in the output folder when building an *OPC UA Client* application.

Instead of referencing the big NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96 an OPC UA Client application can reference the following NuGet packages:

- OPCFoundation.NetStandard.Opc.Ua.Client - version 1.4.371.96,
- OPCFoundation.NetStandard.Opc.Ua.Bindings.Https - version 1.4.371.96.

Note: The OPCFoundation.NetStandard.Opc.Ua.Bindings.Https - version 1.4.371.96 will significantly increase the number of the assemblies from the output folder, therefore if the output folder size is an issue and the HTTPS functionality is not needed it is better to omit adding reference to this NuGet package.

The *Client* will not be able to connect to HTTPS endpoints unless the reference to NuGet OPCFoundation.NetStandard.Opc.Ua.Bindings.Https - version 1.4.371.96 is added to the project.

The *SampleClient* application was modified to reference the *Client and Configuration* NuGet packages. In order to enable HTTPS functionality add reference to OPCFoundation.NetStandard.Opc.Ua.Bindings.Https - version 1.4.371.96.

Note: It is very important to use the same NuGet version for all the NuGet packages from Opc Foundation. The exact NuGet version is specified in the [What's New](#) page and also in the list from above.

The net462 output folder for *SampleClient* will contain:

- 156 files when referencing NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96,
- 17 files when referencing the *Client and Configuration* NuGet packages but not the *Bindings.Https* NuGet package.
- 153 files when referencing the *Client, Configuration* and the *Bindings.Https* NuGet packages.

4 Server API Migration Guide

Starting with version 2.00 of OPC UA .NET Standard SDK we provide a simplified API for server development.

Note: Each application that uses OPC UA .NET Standard SDK version 3.50 must add reference to NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96. The NuGet package contains the OPC UA stack implementation maintained by OPC Foundation under [GIT repository](#).

The steps to migrate to using this API are listed below:

General

1. Copy your solution folder to a new location.
2. Open copy solution in Visual Studio 2022.
3. Add reference to Softing.Opc.Ua.Server.dll to your project.
4. Add reference to NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96. (Or add reference to the fine grained OPC Foundation NuGet packages as described in [Reduce Client's Output Folder Size](#) section).
5. Refactor Server class as described in **section 1** from table below.
6. Refactor node manager classes as described in **section 2** from table below.
7. Refactor the code for starting the server as described in **section 3** from table below.

Id.	OPC UA .NET Standard SDK 1.10	OPC UA .NET Standard SDK 3.50
1	Refactor Server class	
1.1	Server class is derived from <i>Opc.Ua.Server.StandardServer</i> class.	Derive Server class from <i>Softing.Opc.Ua.Server.UaServer</i> class.
1.2	<i>CreateMasterNodeManager()</i> method override	From <i>CreateMasterNodeManager()</i> method override remove any code regarding <i>ShutdownDelay</i> . <i>Shutdown Delay</i> is handled Internally by the OPC UA .NET Standard Stack from OPC Foundation using <i>ShutdownDelay</i> property defined in <i>ApplicationConfiguration</i> . Use that configuration property to manage your server shut down delay.
1.3	<i>OnServerStarted()</i> method override	If your implementation of <i>OnServerStarted()</i> method override does only registration of known variable and object types from current assembly and handles <i>SessionManager.ImpersonateUser</i> event this method can be removed. The registration of variable and object types from current assembly is handled by current implementation of <i>UaServer</i> class. Remove any code that does type registration and use the <i>RegisterTypes()</i> method from <i>UaServer</i> instead. For more details about registering types please read <i>UaServer Class - Register Types</i> section.

Id.	OPC UA .NET Standard SDK 1.10	OPC UA .NET Standard SDK 3.50
		The handling of <i>SessionManager.ImpersonateUser</i> event was replaced by three methods that can be overridden in order to provide authentication: <i>ValidateUserPassword()</i> , <i>ValidateUserCertificate()</i> and <i>ValidateIssuedIdentity()</i> . For more details please read UaServer Class - User Authentication section.
1.4	<i>OnServerStopping()</i> method override	If your implementation of <i>OnServerStopping()</i> method handles only the shutdown delay this method can be removed (see section 1.2)
1.5	<i>LoadServerProperties()</i> method override	<p>Current implementation of <i>LoadServerProperties()</i> method from UaServer creates a new ServerProperties instance using this code:</p> <pre data-bbox="838 798 1556 1353"><code>protected override ServerProperties LoadServerProperties() { ServerProperties properties = new ServerProperties(); properties.ManufacturerName = ManufacturerName; properties.ProductName = Configuration.ApplicationName; properties.ProductUri = Configuration.ProductUri; properties.SoftwareVersion = Utils.GetAssemblySoftwareVersion(); properties.BuildNumber = Utils.GetAssemblyBuildNumber(); properties.BuildDate = Utils.GetAssemblyTimestamp(); return properties; }</code></pre> <p>Override <i>LoadServerProperties()</i> method only if needed, otherwise set the desired properties in configuration.</p>
2	Refactor node manager classes	
2.1	Node manager classes are derived from <i>Opc.Ua.Server.CustomNodeManager2</i> .	Derive node manager classes from <i>Softing.Opc.Ua.Server.NodeManager</i> class.
2.2	<i>CreateAddressSpace()</i> method override	Continue to use <i>CreateAddressSpace()</i> method override to create the static address space. Refactor the code to use the helper methods from OPC UA .NET Standard SDK's <i>NodeManager</i> class for creating object, variable and method nodes as described in Create Static Address Space .

Id.	OPC UA .NET Standard SDK 1.10	OPC UA .NET Standard SDK 3.50
2.3	<i>GetManagerHandle()</i> and <i>ValidateNode()</i> methods override	<i>GetManagerHandle()</i> and <i>ValidateNode()</i> methods implementation from OPC UA .NET Standard SDK's <i>NodeManager</i> class provides desired functionality for all node managers. There is no need to override these methods and they can be removed from your node manager implementation.
2.4	<i>New()</i> method override	Default implementation of <i>New()</i> method provides unique <i>NodeId</i> instances with numeric identifier. Override <i>New()</i> method only if different <i>NodeId</i> creation logic is needed. For more details please read Node Managers - New() method override section.
2.5	Create object, variable and method nodes	Refactor the code to use the helper methods from OPC UA .NET Standard SDK's <i>NodeManager</i> class for creating object, variable and method nodes as described in Create Static Address Space.
3 Start the Server		
3.1	<p>Previous version of <%OEMDATAFEEDDOTNET%> used an instance of <i>ApplicationInstance</i> class from OPC UA .Net Stack to start a server:</p> <pre>ApplicationInstance application = new ApplicationInstance(); application.ApplicationType = ApplicationType.Server; string configurationFile = "SampleServer.Config.xml"; try { // Load the application configuration await application.LoadApplicationConfiguration(configurationFile, false); // Check the application certificate await application.CheckApplicationInstanceCertificate(false, 0); application.ApplicationConfiguration.CertificateValidator.CertificateValidation += CertificateValidator_CertificateValidation; // Start the server await application.Start(new SampleServer()); do { ConsoleKeyInfo key = Console.ReadKey(); if (key.KeyChar == 'q' key.KeyChar == 'x') { break; } } while (true); } catch (Exception e) { Console.WriteLine(e.ToString()); Console.ReadKey(); Environment.Exit(-1); }</pre>	<p>Starting from version 2.00 of the OPC UA .NET Standard SDK, the new <i>UaServer</i> class implementation provides three methods for starting the server:</p> <pre>string configurationFile = "SampleServer.Config.xml"; SampleServer sampleServer = new SampleServer(); try { // Start the server await sampleServer.Start(configurationFile); do { ConsoleKeyInfo key = Console.ReadKey(); if (key.KeyChar == 'q' key.KeyChar == 'x') { break; } } while (true); } catch (Exception e) { Console.WriteLine(e.ToString()); Console.ReadKey(); Environment.Exit(-1); }</pre> <p>For more details about registering types please read UaServer Class - Start the Server section.</p>

Id.	OPC UA .NET Standard SDK 1.10	OPC UA .NET Standard SDK 3.50
	<pre> { break; } } while (true); } catch (Exception e) { Console.WriteLine(e.ToString()); Console.ReadKey(); Environment.Exit(-1); } </pre>	<p>Note: If the CertificationValidation event is handled and one certificate is accepted the certificate is not copied into the <i>trusted</i> folder like it was while using <% OEMDATAFEEDDOTNET%>. The following code shows how to copy the trusted certificate in the <i>trusted</i> folder:</p> <pre> private void CertificateValidator_CertificateValidation(Certif icateValidator sender, CertificateValidationEventArgs e) { try { ICertificateStore certificateStore = m_application.Configuration.SecurityConfiguration .TrustedPeerCertificates.OpenStore(); if (certificateStore != null) { // check for certificate file. var entry = certificateStore.FindByThumbprint(e.Certificate.T humbprint).Result; if (entry == null entry.Count == 0) { certificateStore.Add(e.Certificat e).Wait(); } } catch (Exception ex) { // log exception } } } </pre>

5 Reduce Server's Output Folder Size

The OPC UA .NET Standard SDK is build over the OPC UA .NET Standard Stack from OPC Foundation. The OPC UA .NET Standard Stack from OPC Foundation shall be referenced as NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96 in all *Server* applications. This NuGet reference will bring some unnecessary dlls in the output folder.

Starting with version **2.80** it is possible to trim down the number of assemblies copied in the output folder when building an *OPC UA Server* application.

Instead of referencing the big NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96 an OPC UA Server application can reference the following NuGet packages:

- OPCFoundation.NetStandard.Opc.Ua.Server - version 1.4.371.96,
- OPCFoundation.NetStandard.Opc.Ua.Bindings.Https - version 1.4.371.96.

Note: The OPCFoundation.NetStandard.Opc.Ua.Bindings.Https - version 1.4.371.96 will significantly increase the number of the assemblies from the output folder, therefore if the output folder size is an issue and the HTTPS functionality is not needed it is better to omit adding reference to this NuGet package.

The HTTPS endpoints will not be enabled unless the OPCFoundation.NetStandard.Opc.Ua.Bindings.Https - version 1.4.371.96 is not added to the project.

The *SampleServer* application was modified to reference the *Server* and *Configuration* NuGet packages. In order to enable HTTPS functionality add reference to OPCFoundation.NetStandard.Opc.Ua.Bindings.Https - version 1.4.371.96.

Note: It is very important to use the same NuGet version for all the NuGet packages from Opc Foundation. The exact NuGet version is specified in the [What's New](#) page and also in the list from above.

The net462 output folder for *SampleServer* will contain:

- 162 files when referencing NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96,
- 23 files when referencing the *Server* and *Configuration* NuGet packages but not the *Bindings.Https* NuGet package.
- 158 files when referencing the *Server*, *Configuration* and the *Bindings.Https* NuGet packages.

6 Nugget Packages

The OPC UA .NET Standard SDK is deployed as nuget packages on <https://nuget.org>.

The nuget packages are deployed at each release of OPC UA .NET Standard SDK.

There are 2 nuget packages:

- **Softing.Opc.Ua.Client** - contains the OPC UA .NET Standard SDK Client Library.
- **Softing.Opc.Ua.Server** - contains the OPC UA .NET Standard SDK Server Library.

Note: The nuget can only be used using a Binary License key.

In order to use the nuget packages instead of the dlls from the setup:

- Remove any reference to the OPC UA .NET Standard SDK dlls (Softing.Opc.Ua.Client.dll or Softing.Opc.Ua.Server.dll).
- Open *NuGet Package Manager*.
- From the *Installed* tab uninstall the *NuGet package OPCFoundation.NetStandard.Opc.Ua - version 1.4.371.96*. It will be referenced by Softing SDK nuget.
- Select the *Browse* tab and search for 'Softing'.
- Install the **Softing.Opc.Ua.Client** or **Softing.Opc.Ua.Server** nuget packages version 3.50. Avoid installing both unless your application requires to have both *Server* and *Client* functionality.

Note: Use the latest version, it is released together with the setup of OPC UA .NET Standard SDK version 3.50.

The screenshot shows the NuGet Package Manager interface. On the left, the search results for 'soft' are displayed, listing two packages: 'Softing.Opc.Ua.Client' and 'Softing.Opc.Ua.Server'. Both packages are version 3.0.0.3054 and are published by Softing Industrial Automation GmbH. The descriptions mention the OPC UA .NET Standard SDK Client Library and Server Library respectively. On the right, the detailed view for 'Softing.Opc.Ua.Server' is shown. The 'Install' button is highlighted with a red box. The package details include Version: 3.0.0.3054, Author(s): Softing Industrial Automation GmbH, License: View License, Date published: Tuesday, October 5, 2021 (10/5/2021), Report Abuse: [https://www.nuget.org/packages/Softing.Opc.Ua.Server/3.0.0.3054/ReportAbuse](https://www.nuget.org/packages/Softing.Opc.Ua.Server/3.0.0.3054/), Tags: OPC, UA, Server, Softing, and Dependencies: .NETFramework, Version=v4.6.2 and OPCFoundation.NetStandard.Opc.Ua.Server (>= 1.4.367).

NuGet Package Manager: SampleServer

Package source: nuget.org

Softing.Opc.Ua.Server

Version: Latest stable 3.0.0.3054

Install

Options

Description

This package contains the OPC UA .NET Standard SDK Server Library from Softing.
It is an easy to use API built on top of the OPC UA .NET Standard Stack from OPC Foundation.

Version: 3.0.0.3054

Author(s): Softing Industrial Automation GmbH

License: [View License](#)

Date published: Tuesday, October 5, 2021 (10/5/2021)

Report Abuse: [https://www.nuget.org/packages/Softing.Opc.Ua.Server/3.0.0.3054/ReportAbuse](https://www.nuget.org/packages/Softing.Opc.Ua.Server/3.0.0.3054/)

Tags: OPC, UA, Server, Softing

Dependencies

- .NETFramework, Version=v4.6.2
- OPCFoundation.NetStandard.Opc.Ua.Server (>= 1.4.367)

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

7 Copyrights

Copyright © 2011-2023 Softing Industrial Automation GmbH. All rights reserved.

The Software is subject to the Softing Industrial Automation GmbH's license agreement, which can be found following the link: <https://industrial.softing.com/LA-SDK-en>

The OPC UA .NET Standard SDK uses the following 3rd party libraries:

OPC Foundation:

- **RCL** - .NET Standard Stack is used under the RCL license. The RCL license agreement can be found at <https://opcfoundation.org/license/rcl.html>

Softing is sub licensing the .NET Standard Stack under the RCL license to its OPC UA .NET Standard SDK customers. This allows the OPC UA .NET Standard SDK customers to distribute the .NET Standard.

- **MIT** - The MIT license agreement can be found at <https://opcfoundation.org/license/mit.html>

• **OpenSSL**

The OpenSSL license agreement can be found at <https://www.openssl.org/source/license-openssl-ssleay.txt>

• **Bouncy Castle**

The Bouncy Castle license agreement can be found at <https://www.bouncycastle.org/license.html>