OPC UA SDK
Client + Server Development

Version History

Tested? You want it?

License Model Prices Quotation Order Now

History

2.41.1.0 (released 2024-01-18)

CHANGED

- Updated Microsoft.AspNetCore.Server.Kestrel.Core dependency to version 2.1.25, to fix a vulnerability.

FIXED

- Issue that caused a memory leak in the OpcClient when repeatedly calling OpcClient.BrowseNodes() or accessing file nodes with class OpcFileMethods, OpcFileStream or OpcFileInfo.
- Issue that could cause the OpcClient to incorrectly send Read requests (from the keep-alive timer) even when the ActivateSession call hasn't completed yet.

2.41.0.0 (released 2023-12-02)

CHANGED

- User authentication in the OpcServer is now done only once during ActivateSession, instead of doing it for each request separately.

FIXED

- Issue with OpcVariableNodeInfo.IsArray and OpcVariableNode.IsArray incorrectly returning false (the former since v2.33.0) for an array with 3 or more dimensions.
- Issue with a deadlock occuring when calling OpcClient.Disconnect() or OpcClient.Dispose() while at the same time event subscriptions were active.
- Issue that caused the OpcClient.State not changing to 'Reconnecting' when a connection break was detected while an operation like ReadNodes() was still executing.
- Issue that prevented to create an event filter from an OpcNodeId if it was using a resolved namespace.
- Issue that caused a memory leak when repeatedly creating (and throwing away) OpcMethodNode instances where a Delegate instance is passed.
- Issue that caused reflected type fields to be declared in a wrong order when the type implements an interface.

2.40.0.0 (released 2023-10-23)

NEW

- Introduced new OpcContext.Enter(...) methods to enter into a local context to perform non client / server scenarios.
- Enhanced OpcDataTypeInfo to support ExtensionObjects as system type as well (IsSystemType property).
- Introduced new OpcData.Decode<T>(...) methods to manually decode a binary encoded data stream/buffer to a structured object.
- Introduced explicit support for encoding/decoding/use of ExtensionObjects.
- Improved performance when receiving custom event data for the first time.
- Introduced new OpcNodeSet.Load(string, OpcNodeSetLoadOptions) method to provide more details about the node-set file to load.

FIXED

- Issue with wrong calculated encoding mask when using explicit encoding.
- Issue with encoding/decoding subtypes of built-in data types according to their base types.
- Issue with not reported events at the “Server” node as long as no monitored item has been created to monitor the event expected in its node hierarchy where it resides.
- Missing implicit call to ApplyChanges using OpcEventNode.ReportEvent(…).
- Issues with en-/decoding ExtensionObjects in custom structured data types.
- Issue with nested event type fields like ActiveState/Id not being reported correctly since v2.32.0.
- Issue with mixed data types which are defined using DataTypeDescriptions and DataTypeDefinitions while the definitions uses different field types. The descriptions are now priorized.
- Issue with event subscriptions not working since v2.29.0 when no OpcEventFilter was specified.

2.33.0.0 (released 2023-09-11)

NEW

- Enhanced OpcNodeContext with new NodeManager property which references to the OpcNodeManager in its context an operation is executed.
- Introduced OpcMethodNode.Call(…) method to invoke an implemented method node using a custom context independent from a “client call”.
- Introduced the methods Close(…), GetPosition(…), Open(…), Read(…), SetPosition(…) and Write(…) to invoke the implemented methods of a OpcFileNode using a custom context independent from “client calls”.
- Improved performance of event filter creation using only event types of the node type system provided by the server.
- Introduced support of data types which are only declared using a data type definition and no longer a data type description.
- Introduced support of dynamic union data type resolution and representation using the OpcDataObject.

CHANGED

- OpcContext.Empty.SessionId no longer provides a null reference (Nothing in Visual Basic). Instead it provides OpcNodeId.Null.
- Changed access modifiers of OpcNodeManager.AddNode(…) and OpcNodeManager.RemoveNode(…) methods from protected to public for improved usability.
- Evaluation of OpcVariableNodeInfo.IsArray to no longer use the ValueRank value instead of the ArrayDimensions to decide whether the Value attribute provides an array.

2.32.0.0 (released 2023-08-28)

NEW

- Enhanced DCOM security options with properties to configure AuthenticationService, AuthorizationService and ProxyCapabilities from user code.
- Introduced new Name property in OpcDataTypeAttribute to define a different name per DataType as used by the implementing .NET type.
- Introduced support of lazy initialization of custom OpcData store implementations with support of a default constructor.
- Introduced new IsUnion property in IOpcDataTypeInfo to indicate whether a data type is union data type.

- Introduced new interface IOpcUnion to implement custom union data types using this interface type.
- Introduced typed support of the abstract Number data type which supports the encoding of all primitive numeric data types.
- Introduced new OpcSubscriptionSetup class to create a new subscription with all properties preset in once.
- Introduced new OpcClient.CreateSubscription(...) method which uses a OpcSubscriptionSetup to create a new subscription.

FIXED

- Added a reference to System.Text.Encodings.Web 4.5.1 (for the NuGet package), to fix a vulnerability warning caused by the transitive reference through Microsoft.AspNetCore.Server.Kestrel 2.1.3/1.1.3.
- Issue with not reported events at the “Server” node as long as no monitored item has been created to monitor the event expected in its node hierarchy where it resides.
- Missing implicit call to ApplyChanges using OpcEventNode.ReportEvent(...).
- Issue with always used type identifier of the BaseEventType in the simple attribute operands in an event filter.
- Issue to assure that equal Start and End times when read processed history values do not result into a infinite loop and memory consumption.
- Issues with deadlocks when using UseDynamic and subscriptions consuming structured data types represented using OpcDataObjects.

2.31.0.0 (released 2023-05-17)

CHANGED

- Updated Newtonsoft.Json dependency to version 13.0.3, to fix a vulnerability.
- The UA Wrapper Server for OPC Classic Connections will now always generate a new application certificate, instead of trying to load one from the client's certificate store.
- When no port is specified on a “opc.com:” URL, the port for the classic wrapper server is no longer derived from the classID and progID, but instead a free dynamic port number is determined. This facilitates using multiple OpcClients with the same “opc.com:” URL at the same time.

FIXED

- Issue when updating the OpcServer's ApplicationUri from the selected certificate, which would only be applied for the second start instead of for the first start, which would cause the Reconnect mechanism of the OpcClient to fail.

2.30.0.0 (released 2022-12-07)

NEW

- Added debug-level trace logging when using break monitoring with the OPC Classic UA Wrapper Server.
- Introduced OpcNodeId.Resolution.InheritParentNamespace property to control the inheritance of the namespace of parent nodes to their children.
- The default node identifier resolution now uses the namespace (if not a default namespace) of the parent node to define the “automatic” node identifier of a node (in case OpcNodeId.Resolution.InheritParentNamespace is equals true; the default value).

CHANGED

- The behavior of starting and stopping an OPC Classic UA Wrapper Server is now more deterministic, as each OpcClient uses a dedicated wrapper server, which is now also stopped when the OpcClient disconnects.
- The method OpcCertificateIdentity.Matches() is now virtual to enable custom identity comparison.
- The OpcEvent.EventId property now provides the underlying ByteString instead of an byte-array (before the ByteString was introduced).

FIXED

- Issue with ArgumentOutOfRangeException being thrown when creating an OpcFiniteStateMachineNode without defining states via attributes.
- Issue with NullReferenceException when an OpcFiniteStateMachineNode is created after the OpcServer has already been started.
- OpcEvent.GetValue<T>(string) correctly returns a ByteString value when type parameter T is Opc.UaFx.ByteString.
- Issue with ArgumentException in OpcClient.Connect() when the current AppDomain's friendly name contains non-path characters (e.g. in an Office VSTO add-on).

2.29.0.0 (released 2022-09-16)

NEW

- Added property OpcClientInterfaces.UseBreakMonitoring in order to temporarily stop the UA Wrapper Server for OPC Classic connections while a (DCOM) connection break to the OPC Classic server is detected.
- Introduced OpcNamespace.GetNextId() method to query the next free numerical node identifier from the OpcNamespace.
- Introduced a set of new attributes to specific OpcStateMachineNode specific states and their transitions.
- Introduced new OpcStateNode class to represent the OPC UA StateType.
- Introduced new OpcTransitionNode class to represent the OPC UA TransitionType.
- Introduced support to configure custom OpcFiniteStateMachineNodes using the new state / transition attributes to setup the initial state, the state nodes and the transition nodes using numerical or enumeration values.
- Introduced OpcFiniteStateMachineNode.ChangeStateTo() method to change the current state of the state machine to another state including its validation regarding its existence and translatability.

CHANGED

- OpcNode.AddNotifier is no longer supported on other types than OpcObjectNodes and its subclasses.

FIXED

- Issues with missing data type information for the abstract data type “Structure” (i=22).
- Issues accessing the OpcTransitionVariableNode.VariableId/Node properties.
- Issues accessing the OpcFiniteTransitionVariableNode.VariableId/Node properties.
- Issues accessing the OpcStateVariableNode.VariableId/Node properties.
- Issues accessing the OpcFiniteStateVariableNode.VariableId/Node properties.

2.28.1.0 (released 2022-09-02)

NEW

- Introduced inline initialization of encoding mask related “reserved” fields using a bit-array.

FIXED

- Issue with hostname/IP address not taken into account when generating the port for the OPC Classic UA Wrapper Server.
- Issue with OPC Classic client still calling IOPCBrowseServerAddressSpace.ChangeBrowsePosition() with OPC_BROWSE_TO even though OpcClient.Interfaces.DataAccess.SupportsBrowseTo was set to false.
- Issue with wrong ValueRank and ArrayDimensions in data type definitions of structures using another field to define the length of an array-field.

2.28.0.0 (released 2022-08-19)

NEW

- Enhanced inline definition of DataTypeDictionaryNodes to only provide one in case a data type description of a data type is to be published.
- Unknown (user) defined data types are now resolved using the events OpcData.TypeResolve and OpcClient.TypeResolve using the according encoding identifier and OpcNodeId.Null for the data type identifier.

CHANGED

- Disabling OpcAutomatism.UseDynamicTypeRegistration resulted to a deferred dynamic type registration at the time a custom data type is manually registered.

FIXED

- Issue with ArgumentException whenever the Value property of a OpcStateVariableNode or OpcFiniteStateVariableNode has been changed.
- Issue with OpcData.TypeResolve there the event arguments passed provided the encoding identifier as the type identifier. This has been solved and the encoding identifier is provided as part of the Encoding supplied.
- Issue with OpcClient.TypeResolve not being invoked for every data type.
- Issue with not encoded dynamically resolved data types.
- Issue with missing leading array length in case a data type uses a fixed length field.
- Issues with inline registered data types although OpcAutomatism.UseDynamicTypeRegistration has been disabled.
- Issues with enumerations described in a different data type dictionary as structured data types when using full qualified data type identifiers (value and namespace uri).
- Issue with importing node sets using structure data type definitions without an explicit base type identifier.
- Issue with data type load errors in some third party clients when accessing data type dictionaries only declaring enumerations.
- Issues when selecting an endpoint by constraints, to ignore endpoints that have Mode == None but Algorithm != None and fallback if there is no other endpoint available.

2.27.0.0 (released 2022-06-27)

NEW

- Added new NumberInList property to OpcOrderedObjectNode to determine the current position of a

node within the server.
- Added new OpcClient.TypeResolve event to resolve user defined types according to client / session related conditions.
- Improved OpcDataTypeDictionary.GetType(OpcEncoding) to consider encodings without expanded node identifiers.
- Introduced new protected static (Shared in Visual Basic) OpcNode.DefineModel methods to define node type dependent modelling rules for its whole node model.
- Introduced IOpcDataTypeMapper and its default implementation OpcDataTypeMapper.
- Introduced new OpcData method to register, determine and unregister custom data type mappers.
- Introduced new OpcDataTypeMapper.Convertible and OpcDataType.Delegate methods to construct simple custom data type mappers.

FIXED

- Issue with OpcPropertyNode<T> not being added to its parent in case there the type parameter represents a custom data type.
- Issue with BadNodeIdUnknown on EnumStrings in case an custom enumeration type is defined using EnumStrings.
- Issue of multiple hierarchical references between parent and child nodes in case there a sub-type of the default reference type is used for a child node.
- Regression with enhancement of IOpcNodeInfo.Child overloads and their implementations.

2.26.0.1 (released 2022-04-13)

FIXED

- Regression which disables subclassing of custom data types without data type attributes.

2.26.0.0 (released 2022-04-13)

NEW

- Introduced option to manually define which data types are to be defined using data type definitions and/or data type descriptions using the OpcDataTypeAttribute.
- Added new Declarations property to OpcDataTypeInfo to indicate in which way data type has been declared.
- Added support for arrays of Bit members to define the encoding bits no longer only using scalar encoding bit members.
- Enhanced OPC Watch with new upload button to write a file to a file node (the existing file content is truncated before).
- Introduced new OpcNodeId.Resolution property to configure the ways node identifiers are resolved if no custom node identifiers are used.
- Introduced new OpcNode events BeforeAdd and AfterAdd to handle conditions/states at the time a node is being or has been added to the address space of the server.
- Introduced new OpcNode.Implement method to finalize the node creation before it can be used for the first time.
- Enabled support of implementing standard OPC UA interfaces.
- Introduced OpcOrderedObjectListNode as default implementation of the OrderedListType.
- Introduced abstract OpcOrderedObjectNode as default implementation of the IOrderedObjectType / IOpcOrderedObjectNode.

FIXED

- Issue with not respected custom data type encoding identifiers.
- Issue with cached application certificates using a custom application store path.

2.25.0.0 (released 2022-03-09)

NEW

- Updated OPC UA Stack Types to v1.04.368.
- The server defined data types nodes now do no longer provide a data type definition in case of an opaque data type. Any read request results into BadAttributeIdInvalid and now behaves the same way as built-in opaque data type nodes behave.
- Introduced IOpcData<T> to provide a way to subclass data types which are programmatically not inheritable.
- Different improvements for lazy and cached data type metadata processing to improve performance of encode/decode operations.
- Introduced new OpcClientSecurity related properties to configure the AuthenticationLevel and ImpersonationLevel the client shall use when connecting to an OPC Classic server.

CHANGED

- Default size of (auto) encoding mask from one bit to 32 bit to improve usability of the by default expected 32 bit encoding mask.

FIXED

- Issue with (auto) bit number used when a data type members is marked as optional using OpcDataTypeMemberSwitchAttribute.
- Issue with reloading the servers data type system after a connection has been lost and re-established.

2.24.0.0 (released 2022-03-09)

NEW

- Introduced new IOpcAsyncMehtodCommand interface to implemented asynchronous commands.
- Introduced new default OpcMethodCommands.Execute to provide a custom generic command method to be executed using a command object.
- Introduced new default OpcMethodCommands.ExecuteAsync to provide an async custom generic command method to be executed asynchronously using a command object.
- Introduced support of IOpcAsyncMethodCommand and Delegates pointing to a method which returns a Task or Task<T>. Either are executed asynchronously on method call.
- Introduced new OpcAutomatism.UseAsyncMethodCalls property to globally control the execution of method calls for every server and all its node managers.
- Introduced new OpcNodeManager.UseAsyncMethodCalls property to locally control the execution of method calls of methods defined by the node manager.
- Introduced support for OPC UA NumericRange information using the new OpcRange structure.
- Enhanced OpcReadNode with new Range property to support reading a range of elements in an attribute.
- Enhanced OpcWriteNode with new Range property to support writing a range of elements in an attribute.
- Introduced additional OpcClient.ReadNode method overloads to provide an OpcRange to read a range of elements in an attribute.
- Introduced additional OpcClient.WriteNode method overloads to provide an OpcRange to write a

range of elements in an attribute.

CHANGED

- The OpcNodeAccessToken.IndexRange is now obsolete, use new OpcNodeAccessToken.Range property instead.
- The property type of OpcReadVariableValueContext(<T>).Range has been changed from Ua.NumericRange to OpcRange.
- The property type of OpcWriteVariableValueContext(<T>).Range has been changed from Ua.NumericRange to OpcRange.
- The property type of OpcReadPropertyValueContext(<T>).Range has been changed from Ua.NumericRange to OpcRange.
- The property type of OpcWritePropertyValueContext(<T>).Range has been changed from Ua.NumericRange to OpcRange.

FIXED

- Issues with strict XmlReflectionImporter used in Mono/Xamarin while mapping class information (including expected attributes and property types).

2.23.0.0 (released 2022-03-02)

NEW

- Introduced new Tag property in OpcServiceCommand to assign custom metadata bound to a specific command.
- Introduced Load, Parse and TryParse methods on OpcDataTypeDictionary to provide different ways to consume a XML serialized dictionary in any possible way.
- Overall review of OpcReadHistoryDetails, OpcReadNodeHistoryDetails, OpcReadNodeHistoryRawDetails and OpReadNodeHistoryModifiedDetails.
- Added missing documentation to OpcReadHistoryDetails, OpcReadNodeHistoryDetails, OpcReadNodeHistoryRawDetails and OpReadNodeHistoryModifiedDetails.
- Introduced new ReturnBounds property in OpcReadNodeHistoryDetails.
- Introduced new ReadNode[s]History[Modified] methods to OpcClient to support custom OpcReadHistoryDetails.

FIXED

- OPC Classic connections now specify RPC_C_AUTHN_LEVEL_PKT_INTEGRITY for DCOM in order to handle the hardening changes for CVE-2021-26414.

2.22.0.1 (released 2022-02-25)

FIXED

- Issue calling Acknowledge, Confirm and AddComment on OpcCondition and OpcAcknowledgeableCondition instances.

2.22.0.0 (released 2022-02-11)

NEW

- Introduced new CreateArray method in OpcArrayDimensions.
- Introduced new GetDataType methods in OpcContext to determine context sensitive data type

information.
- Introduced support of OpcModelChangeEvent[Node] and OpcGeneralModelChangeEvent[Node] including its OpcModelChangeItem and OpcModelChangeActions.
- Custom model change event propagation/handling using the new OpcModelChangeEventNode and OpcGeneralModelChangeEventNode.
- Introduced inline event compression of model change events whenever an event snapshot of a OpcGeneralModelChangeEventNode is to be reported.

FIXED

- Issue with wrong definition of ValueRank using OpcArguments constructor accepting custom array dimensions.
- Issue with new DataTypeDefinition when defining a custom enumeration type before it is used for the first time.

2.21.0.0 (released 2022-02-01)

NEW

- Client API - Prepared OPC UA v1.04 compatible support of data type definitions using data type nodes new DataTypeDefinition attribute. This enhancement is backwards compatible to OPC UA v1.03, because data type dictionaries and data type descriptions including the data type encodings are still used to determine declared and described data type information.
- Server API - Introduced OPC UA v1.04 compatible support of data type definitions using data type nodes new DataTypeDefinition attribute. This enhancement is backwards compatible to OPC UA v1.03, because data type dictionaries and data type descriptions including the data type encodings are still used to declare and describe a data type.

2.20.4.0 (released 2022-01-14)

NEW

- Clients are now notified with the status code BadNodeIdUnknown when a node subscribed is deleted later.

2.20.3.0 (released 2022-01-04)

FIXED

- Issue with OpcNamespace.Resolve() and OpcNamespaceCollection.ResolveValue() not working correctly.
- IndexOutOfRangeException when using a non-generic OpcAnalogItemNode in the server.
- Issue with StackOverflowException when (implictly) converting a byte[] as a null reference to ByteString.

2.20.2.0 (released 2021-10-20)

FIXED

- Issue with duplicated string in OpcException.Message.
- Issue with OpcMethodNode using wrong callback target.

2.20.1.0 (released 2021-10-08)

NEW

- Introduced new OpcEnumAttribute to control the definition of the enum members using EnumString or EnumValues. Additionally the OpcEnumAttribute provides properties to control the existing automatisms per enumeration type.

FIXED

- Issue when connecting to a server using 'https:' URLs on .NET Framework.

2.20.0.0 (released 2021-09-17)

NEW

- OpcClient now provides node type information for event, object and variables types.
- OpcNode provides new RegisterType und UnregisterType methods to define/undefine custom node types.
- OpcNode provides new GetType and GetTypeInfo methods to determine Sytem.Type and OpcNodeTypeInfo instances used to define/describe a node type.
- OpcInstanceNode provides new GetNodeValue and SetNodeValue methods to change the Value attribute a dedicated OpcVariableNode which is a child of the instance node.
- Subclassing an OpcObjectNode using the subclass a custom node type definition as well results into inline created property/variable nodes for properties/fields defined in the subclass.
- Subclasses of OpcTypeNode were introduced to define custom node types (OpcNodeType and OpcNodeType<T>).
- Introduced new attributes to describe custom node types (OpcNodeTypeAttribute, OpcNodeTypeIgnoreMemberAttribute, OpcNodeTypeMemberAttribute, OpcNodeTypeMembersAttribute, OpcNodeTypeOptionalMemberAttribute).
- Enhanced OpcNodeTypeSystem class to provide Default and Current node type system related node type information.
- Members in OpcNodeTypeInfo to provide additional information about the type and its children/members.
- Extended data type resolution of OpcArgument to support lazy initialized custom data types as well.

CHANGED

- OpcInstanceNode now determines its DefaultTypeDefinitionId using the node type used to create the instance node.
- OpcMemberOrigins now defines an additional enum member “Method” for method origins.

FIXED

- Issue with lazy defined custom data types used for arguments in OpcMethodNodes.
- Issue with BadNodeIdUnknown while accessing/browsing the Input-/OutputArguments of OpcMethodNodes created before its dedicated OpcNamespace has been initialized and defined in the server (e.g. before OpcNodeManager.CreateNodes has been executed).

2.19.2.0 (released 2021-09-02)

NEW

- Introduced support of IEnumerable<OpcName> in OpcNamePath.

- Introduced new OpcClient.GetType(s) methods to determine any type information associated with a node identifier.
- Introduced new OpcClient.IsSubtypeOf method to determine whether a type is a subtype of another type.
- Introduced new IOpcNodeInfo.Child(OpcNamePath) method to determine a child node using a path of browse names to follow and lookup the according child.
- Introduced new ServiceName property in OpcServiceRequest and OpcServiceResponse.
- Introduced new methods GetResults and GetResultExceptions in OpcServiceResponse to determine request and command related exception information.
- Introduced new OpcServer.RequestJudgement event to handle request issues to decide whether an impeachment (e.g. the request is outdated) is to be dismissed.

CHANGED

- Removed unnecessary new lines in the string representation of OpcException instances.

FIXED

- Misleading use of the same base type (= the top most node type) for the top most base type when examining the type hierarchy of a node type (e.g. BaseType of 'BaseEventType' was 'BaseEventType').
- Possible NullReferenceExceptions using OpcClient.GetDataType and OpcClientGetNodeType methods with a null reference (Nothing in Visual Basic) instead of a valid instance of the OpcNodeId class.
- Issue with too few history values being returned when the server sent no values but a continuation point in one of the responses.

2.19.1.0 (released 2021-09-01)

NEW

- Introduced new OpcEvent.GetValue method for subclasses to determine the correct value according to its event type definition and using its property name as the name of the node its value is to be retrieved. However the same method can be used to resolve the value using the qualified name syntax as known/supported using the DataStore.Get methods. This assures that custom event types do not longer need to know the namespace index in any case.

CHANGED

- **MIGHT BREAK:** IOpcReadOnlyNodeDataStore requires now the implementation of IOpcNamespaceResolver as well.

FIXED

- Issue with resolving the according custom OpcEvent type/instance for a custom event declared using the expanded node identifier of the event type.
- Issue with hangs observed when connecting to OPC Classic servers.

2.19.0.0 (released 2021-08-27)

BREAKING CHANGE: Clients and Servers since v2.17.0.0 are not accessing/providing file nodes according to OPC UA specification. The use a byte array instead of ByteString to read/write file contents. **This issue has been fixed in this version.**

NEW

- Enhanced node type and data type resolution to support expanded (= resolved) node identifiers as well.

CHANGED

- **BREAKING:** OpcFileReadMethodNode.FileReadCallback and OpcFileReadMethodNode.FileReadExCallback now return a ByteString instead of an array of byte.
- **BREAKING:** OpcFileWriteMethodNode.FileWriteCallback and OpcFileWriteMethodNode.FileWriteExCallback now accept a ByteString for the data parameter instead of an array of byte.
- Parameter of OpcFilterOperand.OfType from “value : object” to “typeId : OpcNodeId” to assure that a “OfType” operand is evaluated as expected (= using a OpcNodeId).

FIXED

- **BREAKING:** Regression issue using OpcFileNode access using Read and Write operations after enforcing foundations stack to use ByteString for ByteString and byte[] for byte[] instead of using ByteString for byte[].
- Issue with no history values being returned when server reports status code 'GoodNoData' with a continuation point.
- Issue with a infinite history values being returned when server allows to re-use continuation points.

2.18.5.0 (released 2021-08-13)

NEW

- Introduced support of encoding-based data type retrieval using OpcClient.GetDataType with e.g. OpcValue.DataTypeId.
- Enhanced processing of (structured) data type information to support duplicate data type declarations to redirect to data type descriptions where referenced.
- Simplified property node definition without manually adding the node to the children of a parent node.

FIXED

- Regression in v2.18.4.0 with OpcException+BadServerNotConnected with enabled OpcClient.UseDynamic option while connecting to a server.
- Issues with Siemens OPC UA Servers using a custom server interface setup with custom structured data types. This resolution requires to “enable standard SIMATIC server interface” as well to support the custom server interface setup and its data types used.

2.18.4.0 (released 2021-08-09)

NEW

- Introduced support of Expression<TDelegate> compiled delegates in OpcMethodNodes.
- Introduced indexer using an Int32 to address a dedicated OpcDataField in OpcDataObjects.
- OpcClient.SubscriptionsChanged event is now raised whenever OpcClient.Subscriptions are cleared.
- Removed obsolete code not longer supported in .NET 5.0.
- Improved exception handling/message when importing an UANodeSet.

FIXED

- Issue with late loaded custom type information after the OpcClient.Connected event has been raised.
- Issue with enhanced byte[] support which lead to not longer working sbyte[] support as variable node value (reason of the issue .NET does not differ signed and unsigned types using pattern matching).
- Issue writing byte[] values to variable nodes, because of the foundations stack enforces the use of Variant to consume byte arrays as byte[] instead of a ByteString. However doing so results into wrong type validations writing a variable node using a array of byte as data type / value. Solved that by considering the subsequent stack code accordingly.

2.18.3.0 (released 2021-06-28)

NEW

- Enhanced exception details using inner and aggregate exceptions to populate reasons of OPC Classic connectivity issues when using a OPC Classic to OPC UA wrapper server within the own process.
- Introduded additional OpcNodeIdComparison options: IgnoreNamespaceIndex and IgnoreNamespaceIndexIgnoreCase.

CHANGED

- OpcClassicDiscoveryClient now uses OpcException to represent all stack relating exceptions.

FIXED

- Issues with OpcServerInfo in case of the SDK is used by bundled assemblies.
- Issue when connecting to OPC Classic servers on remote machines.
- Issues with missing nodes while browsing an OPC Classic server with a flat node organization.
- Issue transporting a byte-array to a method node, because of the foundation stack transfers such a value by default as a ByteString instead of using it as it is.

2.18.2.0 (released 2021-06-17)

NEW

- Introduced new Translate methods in OpcServerGlobalization to provide mechanisms to localize custom resources.
- Introduced support of Enum-Arrays which is missed by OPC Stack v1.03.
- Introduced OpcNodeIdComparison and new OpcNodeId.Equals methods to compare node identifiers using different comparison methods.

CHANGED

- OpcNodeId now matches the OPC UA specification and compares node identifiers and namespaces case-sensitive.

FIXED

- Issue with missing preferred locales in OpcSession after activating a session. The locales were only internally updated and used.
- Issues with ServerHalted exceptions when accessing properties of an instance of the OpcServer class before starting the instance.

2.18.1.0 (released 2021-06-02)

NEW

- Added additional setter to OpcBrowseNode.Degree property.
- Introduced new OpcClient.Interfaces property to configure the communication using OPC Classic interfaces.
- OpcClassicDataAccessOptions class provides options to configure the type of organization to use and whether the server supports the BrowseTo service.

FIXED

- Issue when running on Mono and connecting to a OPC UA Server that doesn't send a certificate.
- Issue with missing subscription based notifications after an automatic reconnection.
- Issue with using OpcValue + byte[] which result into a ByteString encoded Byte-Array upon the underlying stack API.
- Issue with encoding the Value attribute of non-generic OpcVariableNodes using an Array of ByteString[].

2.18.0.0 (released 2021-05-26)

NEW

- Improved scenarios where byte[], ByteString and ByteString[] or some of them are used as the data type of a method argument or a variable node.
- Introduced support for OpcDataTypeInfo.Documentation used for the OpcDataTypeNode.Description attribute.
- Introduced new OpcReferenceTypes.GetReferenceTypeName(OpcNodeId) and GetReferenceType(OpcName) methods.
- Completely reviewed and enhanced the types OpcRelativePath and OpcRelativePathElement.
- Introduced support for browse path translation using the TranslateBrowsePath service.
- OpcClient now supports browse path translation using new TranslatePath and TranslatePaths methods.

CHANGED

- Format of ByteString from base64 encoding to hex (= “X2”) encoded / formatted strings which matches foundations binary encoded representation of a ByteString.

FIXED

- Wrong initialized ArrayDimensions and ValueRank attributes of method arguments and variable nodes using byte[], ByteString or ByteString[] or some of them as the data type.
- Wrong initialized DataTypeDictionary node regarding its ArrayDimensions and ValueRank attributes after introducing the new explicit ByteString data type.
- An avoidable NullReferenceException while initializing encodable data types is not longer thrown.
- Issues with wrong determined OpcArgument metadata used to define array-based method arguments.
- Issues with reflecting System.Type information which is defined by-reference e.g. using an “out” parameter in the delegate of a method node.
- Issues with importing UANodeSets its references use aliases to address a dedicated node.
- Issue with missing type identifier whenever inspecting fields of data types there the type of the field is an array type. The array type did not provide the type identifier of its element type.
- Issues in applications with third party components using attribute types which are not loadable. This

fixes TypeLoadExceptions whenever custom data / event types are inspected.

2.17.0.0 (released 2021-05-04)

NEW

- Added support for custom subtypes of the OpcFileInfo class.
- Enhanced OpcMethodContext with Owner information to gain access to the underlying OpcServer instance.
- Enhanced encoding/decoding support for structured data types using their base type information.
- Introduced support of namespaces with null references using OpcNamespace.Create.
- Introduced explicit support of the ByteString data type to simplify differentiation between byte[] and ByteString values.

FIXED

- Issue with servers supplying null references in their array of namespaces.
- Issue with reset OpcNode.Description in case of using e.g. OpcDataVariableNode<T> with a custom data type.
- Issue with unitialized event nodes using in “off-address-space” scenarios e.g. for “lightweight” events.
- Issue with firewall aware and therefore DNS safe host names used to create a new session using the raw discovered endpoint URL instead.
- Issue with TypeInitializationException while testing for Unity in some dynamic assembly load scenarios.
- Issue with InvalidOperationException in foundations BufferManager whenever a still locked buffer is to be returned without a preceding unlock.
- Issue with unresolved data type information in method node arguments using expanded node identifiers for the data types.
- Issue with unresolved data type encoding in the servers type tree using expanded node identifiers for the data type encoding.
- Issue with comparing expanded node identifiers with the same identifier using different namespaces.
- Issue with querying type information from a custom type its name is already in use by a different custom type in a different namespace.
- Issue with NullReferenceExceptions when trying to translate resources without any localized resource information.
- Issue with missing custom data types which are defined among multiple (dynamic) node managers.
- Issue with default reference type inheritance using OpcFolderNodes default reference type Organizes which has not longer been applied to its direct child nodes after the most recent foundations stack update.
- Subsequent issue with declaring/accessing variable nodes using ExpandedNodeId/NodeId as DataType.
- Threading issue when starting multiple OpcServer instances asynchronously.
- Wrong ArrayDimensions attribute value on OpcMethodNodes Input-/OutputArgument nodes.

2.16.1.0 (released 2021-04-21)

NEW

- Enhanced OpcNominalNodeIdFactory to apply custom logic to the OpcName used to define the OpcNodeId created.

FIXED

- Issue with still existing OpcSubscription instances in OpcClient.Subscriptions althougth the client has been disconnected.

2.16.0.0 (released 2021-04-20)

NEW

- Introduced support for custom OpcFileNode sources using the new IOpcFileInfo interface.

FIXED

- Issue with declaring/accessing variable nodes using ExpandedNodeId as DataType.
- Issue with FileNotFoundException using some third party assemblies while dynamic type registration is enabled.

2.15.0.0 (released 2021-03-31)

NEW

- Introduced new Opc.UaFx.Client.OpcSubscription.Client property to get access to the client which is the owner of a subscription.
- Introduced new OpcServer.Binding.Addresses property to define explicit IP addresses of the network interfaces to bind to and listen on.
- Introduced OpcNodeCollection.TryGet method to try to look up a node using its node identifier.
- Introduced OpcNodeManager.GetNode methods to query nodes using their node identifier or a path which represents a slash ('/') combined sequence of browse and/or symbolic names to describe the path to the node to determine.

FIXED

- Issue with creating property nodes without a parent node.

2.14.0.0 (released 2021-03-04)

NEW

- Introduced new static OpcName.IsNullOrEmpty method which ignores the namespace information associated with the OpcName.
- Introduced new static OpcNodeId.IsNullOrEmpty method which ignores the namespace information associated with the OpcNodeId.
- Introduced new OpcNamespace.GetId(object) method to determine a new node identifier without to know which type of value it is.
- Introduced dynamic resolution of data type information when using not registered data types.
- Introduced additional non-generic OpcData.RegisterType methods to not longer enfore the use of the generic type parameter.
- Introduced new OpcDataTypeNode.Create(...) methods to provide a non-generic way to initialize new OpcDataTypeNode instances.

CHANGED

- Removed duplicate references to DataTypeNodes when using the OpcAddressSpace class to describe the nodes of the server.

FIXED

- Issue with “malformed” or “TooMany” subscriptions preventing the OpcClient to automatically reconnect after connection loss.
- Issue with inline initialized OpcDataTypeNode using new dynamic resolution of data type information.
- Issue with missing custom enumeration type references in the global data type system.

2.13.0.0 (released 2021-03-01)

NEW

- Improved dynamic resolution of the data type attribute supporting expanded node identifiers as well.
- Improved dynamic data type encoding / decoding to support subclassing and cascaded custom structured data types.
- Introduced new OpcDataTypeIgnoreMemberAttribute to explictly exclude a member from being encoded or decoded.
- Introduced new OpcDataTypeAttribute.NamespaceUri and OpcDataTypeAttribute.UseDataTypeDescription properties to define the target namespace used for the data type dictionary of the data type and to define whether the data type needs to be described in a custom data type dictionary.
- Introduced support of custom target namespaces for custom enumeration types.
- Introduced support for embedded custom data types when using dynamic data exchange using container types like Variant and/or ExtensionObject.

FIXED

- Issue with missing target namespace reference in data type dictionary when using custom data types without target namespace information.

2.12.3.0 (released 2021-02-17)

NEW

- Enhanced support of IL2CPP regarding issues with embedded string evaluation.

FIXED

- Issues using default certificate probing paths in Unity using UWP.
- Issue with calling method nodes after introduction of argument localization.

2.12.2.0 (released 2021-02-15)

NEW

- Implemented localization support for arguments of a method node.

FIXED

- Issue with missing localized information in case there a client uses the locale of a neutral culture.
- Issue with null references (Nothing in Visual Basic) in case of reading localizable attributes without localization information.

2.12.1.0 (released 2021-02-11)

NEW

- Introduced new OpcNodeExportOptions to setup the information exported while exporting nodes to a node set.
- OpcNodeManager.ExportNodes(OpcNodeExportOptions) method overload to specify custom node export options.
- Introduced new OpcNodeGlobalization class to maintain the resources used to localize the attributes of a dedicated node.
- OpcNode.DisplayNames and OpcNode.Descriptions properties to provide an instance to control the localization on node level per localizable attribute.

CHANGED

- Bidirectional references between DataTypeDescription-, DataTypeEncoding- and DataType-Nodes are now unidirectional (using default export options).

FIXED

- Issue with automatically determined server information using the process related version information including an evaluation of the ProductVersion and the FileVersion.
- Issue with overridden OpcNode.SymbolicName in case the node has been imported with a different SymbolicName as the DisplayName.
- Issue with not longer multi-lined base64 encoded data type dictionary (using default export options).
- Issue with removed ParentNodeId attributes in NodeSets (using default export options).
- Issue with re-ordered nodes (using their NodeId) in exported NodeSets. Now they are exported in the order of their definition / implementation.
- Issue with missing super types while importing nodes once exported to a NodeSet (because of re-ordered nodes).

2.12.0.0 (released 2021-02-04)

NEW

- Introduced new Namespaces property in IOpcApplicationInstance and OpcApplicationInstance to provide the namespaces used.
- Introduced new Namespaces property in OpcClient to reflect the namespaces defined by the server connected to.
- Introduced new Namespaces property in OpcServer to provide the namespaces used by the node managers of the server to organize the address space.
- Introduced new Namespaces property in OpcContext to provide the namespaces currently valid within a given context.
- Implemented IOpcNamespaceResolver in OpcNamespaceCollection and OpcReadOnlyNamespaceCollection.
- Resolve method in IOpcNamespaceResolver to resolve another (= “online”) namespace using the metadata from a (offline / local) defined namespace.
- OpcNamespace now provides an Uri whenever the OpcNamespace.Value is a valid Uri.
- Connections to OPC Classic servers through an inline used wrapper server are now secured using the same user identity setup in the OpcClient.Security.UserIdentity.
- OpcClient and OpcClassicDiscoveryClient now support the use of a host identity for DCOM authentication. Note: In case of the OpcClient the OpcClient.Security.UserIdentity is used from DCOM authentication as well.

- Introduced new Scope property in OpcNamespace to describe the context within the namespace has been created and its metadata applies to.
- Improved performance initializing OpcNodes during the startup of the OpcServer.
- Using OpcNodeManager.ctor(OpcNamespace, ...) with a namespace which references to an explicit namespace index results into an ArgumentException, because of the server will define the namespace indexes associated with the different namespaces used.
- The OpcNamespace instances in the OpcNodeManager are now synchronized with the OpcNamespace instances in the new OpcServer.Namespaces property. Note that until the server is about to start the instances in OpcNodeManager.Namespaces might be exchanged with the instances in the OpcServer.Namespaces property. In fact in e.g. OpcNodeManager.CreateNodes(...) all OpcNodeManager.Namespaces are already in-sync with the OpcServer.Namespaces.

CHANGED

- The OpcNamespace.Default namespace now provides the default Value and Uri used to identify the foundations default namespace.
- Removed obsolete methods from IOpcNamespaceResolver and all its implementations in OpcClient, OpcServer, OpcContext and OpcNodeManager.
- OpcClient.State transitions to not longer change the state (using the by default activated reconnect mechanism) by calling e.g. ReadNode while the OpcClient is about re-establish a broken connection.
- In case of a broken connection the OpcClient tries (by default) to re-establish the connection and re-use the previously used session. If this works the OpcClient will raise the Reconnected event as before. In case the client has been forced to create a new session (e.g. upon timeout of the previous session) the OpcClient will now raise the Connected event instead of the Reconnected event to provide a way to differ these conditions in user code.
- Comparison in OpcNodeId and OpcNamespace to provide results typically used by .NET API when comparing to a null reference (this differs to the implementation used by the foundation).

FIXED

- Issue with “collapsed” endpoint urls when discovering a server which returns e.g. "opc.tcp://:4840".
- Issue with un-resolved node identifiers used when manually creating monitored items being added to a subscription.
- Issue with “over-resolved” node identifiers through resolving a node identifier always to a new node identifier instead of overriding namespace information in existing node identifiers.
- Issue in OPC Classic Stack with growing number of threads in case of connection loss while communicating with OPC Classic Servers.
- Issue with providing custom data types without to define an explicit NamespaceUri in its DataTypeEncoding.
- Issue with lost OpcNamespace information after inline re-initializing the node identifier of a node its foundation node identifier has been changed in meantime.
- Issue in applications where multiple servers are accessed or a client and server are implemented in the same process; however all involved namespaces did not match every use case. This has now been solved through not longer inline resolving the namespace of an existing node identifier and OpcNamespace instance. If this is desired the user now have to resolve the namespace on its own.

2.11.5.0 (released 2020-12-21)

NEW

- Optimized usage of aliases in exported UA NodeSets (unused are now removed from the UA NodeSets generated).

- Removed redundant node information by comparing DisplayName and Description. In case both are equals, the Description attribute is not longer exported to the UA NodeSet generated.
- Removed redundant ParentNodeId attribute from the UA NodeSets generated in case there the parent node is already referenced using the ReferenceTypeId of the child node.

FIXED

- Issue with NullReferenceException in OpcServer in case of trying to add a new node (through a Client) with an unknown / invalid parent node identifier.
- Issue with missing node references (to nodes which are defined within a different node manager) in UA NodeSets when exporting nodes its node identifier has been implicitly assigned.
- Issue with exporting nodes which refer to each other using the ParentNodeId attribute and an explicit node reference (the ParentNodeId is now omitted in such a case; if child.ReferenceTypeId [= the default reference type between parent and child nodes] is equals to the reference to declare in the UA NodeSet). Importing the resulting UA NodeSets led to an address space with multiple references between the nodes (which were defined by the ParentNodeId attribute and the references defined).

2.11.4.0 (released 2020-12-15)

NEW

- Introduced new OpcContext.Owner property which refers to an IOpcApplicationInstance like OpcClient or OpcServer the context belongs to (if available).
- Introduced new OpcException.Context property which offers an instance of the OpcContext class and therefore provides additional context-sensitive information regarding an exception thrown/caught.

FIXED

- Issue using abstract methods as delegate for method nodes (caused by an issue in the .NET Framework using GetCustomAttribute<T> extension methods).

2.11.3.1 (released 2020-11-27)

FIXED

- Issue connecting to Servers evaluating endpoint URLs using case sensitive comparsion and not supporting '/' terminated URLs.

2.11.3.0 (released 2020-11-23)

NEW

- Added missing OpcEventDataSetItem.ClientID property.
- Added missing code documentation of OpcSecurityPolicy, OpcSecurityMode and OpcSecurityAlgorithm.
- Introduced new PreferredLocales property to OpcContext, OpcOperationContext and OpcSession.
- Introduced new OpcNode.Read/WriteDisplayNameCallback property.
- Introduced new OpcServer.GetSession(sessionId) method to query a specific session using its session identifier.
- Introduced new OpcServer.Globalization property to provide a more advanced way to localize data and information.

CHANGED

- Used data type in OpcNode.Read/WriteDescriptionCallback property from String to OpcText.
- OpcDataChangeDataSetItem.ClientID from Int32 to Int64 to match used type in OpcMonitoredItem.ClientID.

FIXED

- Issue with missing OpcNotification.Data using OpcNotification.GetDataChanges() without accessing OpcNotification.Data before.
- Issue with type initialization in OpcClient using Unity with IL2CPP which results into a different type initilization order.

2.11.2.0 (released 2020-11-10)

NEW

- Experimental OpcNodeManager.ExportNodes method to generate OpcNodeSet instances from the address space defined by a node manager. Use the method there required and send us feedback in case of any issues or desired enhancements.

FIXED

- Issue that could lead to a hang in OpcClient.Dispose() when a subscription was active.
- Possibility of Exceptions in OpcClient.Dispose and OpcServer.Dispose.
- Issue validating license information using IL2CPP and .NET Standard 2.0 as scripting backend in Unity.
- Issue which prevented use of an existing SynchronizationContext to process notifications in the same context a subscription was created from.
- Issue adding nodes with additional node references (references were not added to the address space).

2.11.1.0 (released 2020-11-05)

NEW

- The OpcServer is now able to get stopped while it is in suspended state.
- Property to configure the TimestampsToReturn for all requests using OpcClient.Services.ReadNodesHistory.TimestampsToReturn.
- Property to query the ApplicationUri contained in the subject alternate name extension of certificates using the OpcCertificateInfo.ApplicationUri property.
- Property OpcClientSecurity.AutoUpgradeEndpointPolicy to enable endpoint selection using a match (= default) and next-best technique to determine an endpoint which at least satisfies the policy configured.

CHANGED

- The constructor of OpcCertificateJudgementEventArgs does not longer verify if the specified certificates are a null reference (Nothing in Visual Basic), because of some Server might supply no certificate data.

FIXED

- Issue with not longer stored last notifications in subscriptions / monitored items if they are not (longer) created for the current session.

- Issue with missing security policies after restarting the OpcServer.

2.11.0.0 (released 2020-10-06)

NEW

- Introduced method OpcServer.Suspend() to change the servers state to Suspended.
- Introduced method OpcServer.Resume() to change the servers state back to Started.
- Enumeration members OpcServerState.Suspending and OpcServerState.Suspended.
- Enhanced OpcSession.Close to not longer only discard session related diagnostic data. Instead this method now really closes the session.
- Added missing code documentation to subscription related types in the Client namespace.

CHANGED

- OpcSession.Close closes the session and invalidates the session-related identifier.
- OpcClient methods validate the connection state using an OpcException (with BadServerNotConnected) instead of using an InvalidOperationException.
- Default QueueSize of OpcMonitoredItem created using OpcClient.SubscribeEvent(s) from int.MaxValue to uint.MaxValue to enfore that the server uses the maximum queue size supported for event notifications (which was initially intended with the wrong int.MaxValue).
- Renamed OpcMonitoredItemStatus.DataItemQueueSize to QueueSize to match the naming used in OpcMonitoredItem (which defines the requested size while the status gives the revised size).
- Returned type of OpcMonitoredItemStatus.Id from Int32 to Int64 to cover the underlying non-CLS compliant UInt32 of the foundations stack.
- Adopted recently reviewed API of OpcMonitoredItem to OpcMonitoredItemStatus and reworked OpcMonitoredItemStatus.AutoFlushCache to QueueMode (as in OpcMonitoredItem).

FIXED

- Wrong status code used for communication errors (BadNotConnected → BadCommunicationError).
- Client connected to a fallback/unsecure endpoint in case of a preset SecurityPolicy is not supported by a Server.
- Issue with not applied OpcSubscription.TimestampsToReturn without changing the monitored items of a subscription.
- Typos in RegisterNode(s) code documentation.

2.10.0.3 (released 2020-09-11)

FIXED

- Issue connecting to OPC Classic servers.

2.10.0.2 (released 2020-09-09)

FIXED

- Issue with comparing OpcNodeIds using a Byte-Array as node identifier value.

2.10.0.1 (released 2020-07-15)

NEW

- Updated NuGet package description.

2.10.0.0 (released 2020-07-14)

NEW

- Enabled auto-initialization-features of nodes added using the add nodes service. This results into the same behaviour as used when adding a node within a custom node manager.
- Enhanced OpcCertificateManager.SaveCertificate method to determine format of certificate using the file extension of the specified filePath parameter.
- Introduced OpcValueRange.Of methods to dynamically determine the according OpcValueRange for a specific type of data.
- Non-generic OpcAnalogItemNodes can now define the according OpcValueRanges for the InstrumentRange and EngineeringUnitRange on their own as well (using the DataType attribute).
- Added support to encode/decode System.Decimal (not supported by OPC UA) using System.Double. This applies to fields in data structures.
- ProductUri property in OpcClient and OpcServer to simplify the configuration of it.
- Added automatic initialization of ProductUri information in Client and Server applications.
- Introduced use/support of an existing SynchronizationContext to process notifications in the same context a subscription was created from.
- Improved developer expierence in GUI applications (like Windows Forms) regarding correct multi-threading in GUI applications to handle received notifications.
- Introduced OpcNode.UpdateChanges methods to notify about changes on behalf of a node (and its children).
- Added missing documentation to different types.

CHANGED

- OpcNode.OnAfterApplyChanges and OpcNode.OnBeforeApplyChanges using OpcNodeChangesEventArgs instead of EventArgs.
- OpcNode.AfterApplyChanges and OpcNode.BeforeApplyChanges pass OpcNodeChangesEventArgs instead of EventArgs to event handlers.

FIXED

- Issue writing an analog item node which is offered by a custom node manager without to provide an instrumental range to validate values written to the node.
- Issue with only once updated SourceTimestamp in OpcValues (if there was not a SourceTimestamp explictly defined) written to the variable node.
- Issue in OpcNodeHistorian in case there a nodeset imported node with enabled historizing is attached to a new OpcNodeHistorian while the node is missing HistoryConfiguration and/or AggregateConfiguration nodes.
- Issue linking assembly in C++ CLR projects using C++/CLI which results into C2686.
- Issue with NullReferenceException using .NET Core 3.1 while receiving event notification data.
- Issue with not expanded node identifiers in namespace index one (reserved for the core/application namespace).

2.9.2.1 (released 2020-05-08)

FIXED

- Issue reading a structured data type which uses an array as a field while OpcClient.UseDynamic is equals true.

2.9.2.0 (released 2020-05-06)

NEW

- Added additional constructor overloads to OpcArgument to support the use of OpcDataType, the identifier of a data type and custom array dimensions as well.
- Added support for n-dimensional (= multi-dimensional) arrays of structured data types.
- Added support for n-dimensional variable nodes (this includes new ArrayDimensions property on OpcVariableNode as well).
- Added support of null references as field values for fields using a reference type as the type of value.
- Added support of the ObjectType NodeClass in calls to the add nodes service.
- Added support of the VariableType NodeClass in calls to the add nodes service.

FIXED

- Issue with automatic definition of value rank of variable nodes in case of n-dimensional custom structured types.
- Issues in stack to support n-dimensional custom structured types.
- Issue loading UANodeSets with extended values for AccessLevel and UserAccessLevel in UAVariable entries (belongs to stack v1.04).
- Issue with NullReferenceException in OpcDataTypeInfo.BaseType in case of data types defined in a node set using an UATypeDefinitions without any field.
- Issue with NullReferenceException in case of an encoding-independent data type (= enum) is defined in the address space. This issue belongs to the server API only.
- Issue with wrong determined value rank in case of System.Byte[], because of its was matched with the ByteString by the foundations stack.

2.9.1.0 (released 2020-04-22)

FIXED

- Issue which led to duplicate browsed references between parent/child nodes imported using a nodeset.
- Issues with newer versions of System.ServiceModel.Primitives. Rolled back dependency to v4.5.3 in all target frameworks.
- Issues with subsequent evaluation licenses used for bundle licensing (client + server).
- Issue with wrong determined value rank in case of System.Byte[], because of its was matched with the ByteString by the foundations stack.
- Issue with result of OpcNamespace.Get(int, Uri) which result into a OpcNamespace.Value which did not express the Uri.OriginalString as defined by the namespaceUri parameter.
- Issue with cascaded custom data types if they do not define a custom NamespaceUri in their DataTypeEncoding. This issue belongs to the server API only.
- Issue with not applied BrowseName in AddNodes in case of type node classes. This issue belongs to the server API only.
- Issues with type node classes regarding wrong add nodes command validation. This issue belongs to the server API only.

2.9.0.0 (released 2020-04-01)

NEW

- Unified product licensing which does not longer differ between frameworks.

- All Licenser classes now offer different FailIf... and ThrowIf... methods to verify the used license on-demand by user code. Use FailIf... methods in your DEBUG configurations and use ThrowIf... methods in your RELEASE configurations to protect you from the unlicensed use of the SDK.
- Added new GetNodeType(...) methods to OpcClient (only for event node types).
- Added new GetNodeTypeSystem(...) methods to OpcClient (only for event node types).
- Added new GetNodeType(...) methods to OpcNodeSet.
- Added new GetNodeTypeSystem(...) method to OpcNodeSet.
- OpcMethodNode now supports custom defined input and output arguments when using IOpcMethodCommand.
- Added new GetNodeType(...) methods and GetNodeTypeSystem() method to OpcNodeSet.
- OpcNode now inherits the OpcNode.Namespace to its child nodes (in case they use the default namespace determined by their OpcNode.Id).
- Introduced new specializations of OpcTypeNodeInfo: OpcObjectTypeNodeInfo and OpcVariableTypeNodeInfo (subclassing OpcTypeNodeInfo).
- Improved expierence while debugging and inspecting the properties of OpcNodeInfo instances (and its subclasses).
- Browsing supports now to read additional attributes besides the attributes supported by the browse services.
- OpcNodeInfo now implements IOpcNodeInfo to provide a more fluent use of nodes instances independent from their source.
- Additional configurable services in OpcClientServices: Browse and ReadNodes.
- Significantly improved performance when constructing event filters.
- IOpcNodeInfo now defines the additional methods: AttributeValue(...), Child(...) and Children().
- Unlimited browse, read and write operations upon automatic request partitioning (OpcClient).
- OpcClient.CertificateJudgement event to handle certificate issues after a connection has been established and certificate information has been successfully validated first.
- OpcModel representing the model information in node sets.
- OpcMemberSwitch.Conditions property to inform about the condition under which a switch is evaluated.
- Direct support of .NET Core 3.1 as one of our target frameworks.
- Implemented ReplaceNode method in OpcNodeManager to replace an existing node recursively in Nodes, Notifiers and MonitoredItems.
- Added additional metadata properties (LastModified and Name) to OpcNodeSet.
- A general implementation of the IOpcMethodCommand interface using callbacks: OpcMethodDelegateCommand.
- Added missing documentation to collection and some other types.
- OpcNodeSetManager.ImplementNodeCallback to implement nodes just using a simple callback method.
- OpcData.GetNodeId now supports array types as well.
- Added encoding information in address space of server (internal).
- Generic implementation of IOpcMethodCommand for delegates OpcMethodDelegateCommand.
- Support of System.Object and System.Object[] for encoding / decoding data types.
- Support of Serialization / Deserialization attributes defined in System.Runtime.Serialization on data types.
- OpcDataTypeMembersAttribute to filter members which shall be encoded / decoded on data types.
- OpcDictionary<TKey, TValue> as a bare and not by default available data type to transport dictionaries using OPC UA.
- It is not longer necessary to link the name nor identifier of custom defined root nodes returned by OpcNodeManager.CreateNodes(...) manually to the default namespace of the custom node

manager.
- Respecting order of fields as they are declared within their inheritance hierarchy to always define the fields of the base data type first.
- Added new GetDataType(…) methods to OpcClient.
- Added new GetDataType(…) methods to OpcNodeSet.
- IOpcNodeInfo now defines the methods Child(…) and Children() to query the child nodes of a node instance.
- Implemented new OpcNamespaceCollection.
- OpcNodeSet.Namespaces property to provide the namespaces referred to in a node set.
- IOpcMethodCommand to define/use custom command instances instead of being forced to define an explicit delegate.
- OpcMethodCommands class to provide predefined commands such as NotImplemented.
- Enhanced OpcMethodNode to support IOpcMethodCommand instances as well as invocation target instead of using a delegate.
- Enhanced generic OpcDataTypeNode<T> class to support not longer only enumeration types. The class now supports the use of class and struct types as well and maps them to OPC UA structures.
- Introduced new abstract and non-generic OpcDataTypeNode class.
- Static OpcContext.Empty property to define an empty context instance where required and supported.
- Implemented IOpcNamespaceResolver in OpcContext to resolve namespaces of OpcNodeId instances.
- OpcTypeSystem class which acts as the new base class of the existing OpcDataTypeSystem and the new OpcNodeTypeSystem classes.
- OpcDataTypeDictionary.Documentation property.
- OpcNodeSetManager class which enables simple hosting of OpcNodeSets using OpcNodeSetManager.Create(…).
- Protected virtual methods OpcNodeManager.ImportNodes() and OpcNodeManager.ImplementNode(IOpcNode).
- Unified and simplified DataType resolution especially using non-built-in data types for variable and property nodes. This affects the nodes: OpcDataItemNode, OpcDataItemNode<T>, OpcAnalogItemNode, OpcAnalogItemNode<T>, OpcDataVariableNode, OpcDataVariableNode<T>, OpcConditionVariableNode, OpcConditionVariableNode<T>, OpcPropertyNode, OpcPropertyNode<T> and OpcVariableNode.

CHANGED

- **License codes used prior v2.9 do not longer work. Customers will receive new license keys working in all frameworks supported.**
- OpcAttributeWriteAccess is now based on UInt32.
- The default value of OpcAutomatism.AlwaysParseStringIdentifiers is now false. As a result, the value is no longer automatically parsed when the constructor `OpcNodeId(string)` constructor is used. For more details see OpcAutomatism.AlwaysParseStringIdentifiers
- Moved OpcReadOnlyNamespaceCollection from namespace Opc.UaFx.Server to namespace Opc.UaFx.
- Default value of DataType attribute of OpcDataTypeDictionaryNode from Byte to ByteString to match definition of specification.
- Renamed Description property of IOpcDataTypeInfo, OpcDataTypeInfo and OpcDataFieldInfo to Documentation to match the definition of the OPC UA specification.
- OpcEncoding.All(…) now constructs the OpcEncoding.Id using the default browse names for the different encoding types.

- OpcNamespace.ToString() now always returns a string which just represents the OpcNamespace.Value + OpcNamespace.Index (if not zero).
- OpcNodeSet.GetDataTypeSystem() now provides a OpcDataTypeSystem instance which uses the typical browse name for the type of encoding used by the type system.
- Removed method IOpcTypeProvider.GetType(OpcEncoding), because of encoding information only applies to data types.
- OpcTypeCategory.EventType and OpcTypeCategory.ReferenceType has been removed.

FIXED

- Issues with validating outdated certificates or certificates which violate some policy regarding general valid certificates. Such certificates can now handled by the CertificateValidationFailed event.
- Issue with the automatic application configuration in case there the entry assemblies AssemblyTitleAttribute is not present or defines a string.Empty as assembly title.
- Issue with variable node values changing their value using different complex data types (expressed using ExtensionObject as the data type of the node).
- Wrong encoding mask in case there is more than one optional field declared in a data type.
- Wrong interpretation in OpcArgument.IsArray and OpcArgumentInfo.IsArray.
- An OpcVariableNode.Timestamp initially defined is not longer reset during address space creation.
- Issues with wrong determined default value ranks and default data type identifiers using OpcDataTypes.GetValueRank(..) and OpcDataTypes.GetNodeId(...).
- Issue with DataTypeDictionary nodes which do not define any DataTypeDescription nodes while constructing the DataTypeSystem using OpcClient.GetDataTypeSystem().
- Issues when decoding structured data using the new Opc.UaFx.Bit.
- Issue using OpcNodeId.UriIdentifier(...) in case there the OpcNodeId.Value is not of the type System.String.
- OpcNamespace now preserves the original string used to define the OpcNamespace without any subsequent changes as the value of the OpcNamespace.Value property.
- Some conditions under which a newly created OpcNode without a name could not be renamed later.

2.8.3.1 (released 2020-01-24)

FIXED

- Issue with not longer working subscription-handlers after a long running reconnection attempt succeeded.
- Issue with disposed socket in diposed TcpMessageSocket used in a request header representing the last request send to the server to which the connection got lost.

2.8.3.0 (released 2020-01-16)

NEW

- Support of OpcExceptions in delegates used to define method nodes. This enables the definition of bad results without the need to manually change the result of the method call using the OpcMethodContext-Parameter.
- Added support for additional access levels (StatusWrite and TimestampWrite).
- Added missing documentation to OpcAccessLevel, OpcVariableNode.AccessLevel and OpcVariableNode.UserAccessLevel.
- Added support to disconnect and abort a pending reconnect attempt on OpcClient.
- Changes to improve the experience with UWP applications regarding the use of the native tool chain.

CHANGED

- Reviewed OpcClient.State changes in order to detect a lost connection to the server.
- OpcClient.State now changes automatically only to Disconnected (during a running request) if ReconnectTimeout is greater than zero (reconnection is manually handled).

FIXED

- Issue with not applied status (code) information passed to one of the constructors defined by OpcValue.
- Issue with TypeDictionaries which do not offer NamespaceUri information.

2.8.2.1 (released 2019-12-13)

FIXED

- Issue with parallel subscription publishing while calling a method node in OpcClient.
- Issue with not updated OpcVariableNode.Value in case there is no custom WriteVariableValueCallback associated with a variable node.
- Issue with conditions upon the OpcClientKeepAlive.Interval is not applied to the current session.

2.8.2.0 (released 2019-11-06)

NEW

- Introduced new Opc.UaFx.Bit as a .NET-based representation of the foundation-defined “Bit” value type for which no concrete implementation exists so far.

CHANGED

- Explictly encoded information known as the “EncodingMask” of a structured data type is now handled using the new Opc.UaFx.Bit type instead of System.Boolean.

FIXED

- Issue with mixed-structured data types using leading boolean values followed by additional non-boolean values which sometimes resulted into wrong encoded data streams.
- Issue connecting to OPC Classic servers using OPC Watch without a custom application certificate.
- Issue in OpcServerInfo.FromProcess(…) in case there the ProductVersion of the FileVersionInfo of the ProcessModule of Process.MainModule provides a version string which not only consists of numbers and dots like '1.2.3.4-preview'.

2.8.1.3 (released 2019-10-24)

FIXED

- Issue with exchanging structured data types without using the upcoming metadata-support.

2.8.1.2 (released 2019-10-23)

CHANGED

- iconUrl and licenseUrl information in NuGet packages upon their deprecation.

FIXED

- Issue with not longer working OPC Classic support (An internal error occurred as a result of a programming or configuration error).
- Issue with wrong interpretation of Int32 as return type of method nodes.

2.8.1.1 (released 2019-10-11)

NEW

- Implemented support for method parameters using “System.Object” which are transported as OPC UA variant values.

FIXED

- Issues with disregaded method parameters of the type “System.Object”.
- Issue with reading large file nodes (representing files their content exceeds the MaxByteStringLength).

2.8.1.0 (released 2019-09-25)

NEW

- Implemented RegisterNodes and UnregisterNodes methods on the OpcClient that allow to register nodes on the OPC UA server for the current session for possibly faster access.
- Enabled OPC Classic API in .NET Standard target to support the use of the API in .NET Core applications running on Windows.
- Implemented OpcDataObject.HasField(name) and OpcDataObject.TryGetField(name, out field).
- Improved performance by excluding foundation defined data type dictionaries while querying the data type system of the server.
- Introduced new OpcClient.UseDynamic property and changed default value of OpcAutomatism.UseDynamic to the value false to improve the performance while connecting without the need of the knowledge of data types.
- Implemented support for servers which do not support range-based reads to query the data type dictionary.

FIXED

- Crash caused by an exception in Socket.Shutdown() that was not caught.

CHANGED

- OpcAutomatism.UseDynamic property default value is now false. Change its value or the OpcClient.UseDynamic to true to enable use of OpcDataObject.

2.8.0.0 (released 2019-09-18)

NEW

- Introduced a whole set of new classes to support the definition, consumption and reflection of custom data types in client applications.
- Introduced new OpcNodeSet class to provide a file, string or stream based source to offer data type system information from a UANodeSet.
- Implemented new OpcClient.NodeSet property to assign an instance of the OpcNodeSet class which is to be used to query data type system information instead of querying the servers data type system.

- Implemented new OpcClient.GetDataTypeSystem() method which determines the data type system which is used by the client to resolve data type information.
- Introduced new OpcAutomatism.UseDynamicTypeRegistration property to control the automatic type reflection used by client and server applications to determine the different data and event types declared.
- Introduced new OpcAutomatism.UseDynamic property to control the dynamic type resolution while consuming custom typed but not explictly declared data types in client applications.
- Introduced new OpcNodeValue<T> class, OpcAttributeValue<T> and IOpcNodeAttribute with IOpcNodeDescriptor in conjunction with OpcNodeValuePair to declare “node-bound” values.
- Enhanced OpcClient class with new ReadNodeValue and WriteNodeValue methods to read and write “node-bound” values.
- Improved Exception handling to preserve StackTrace information from user code down to foundation and communication layer.
- Implemented new OpcClient.KeepAlive property and the according OpcClientKeepAlive class to control and monitor the keep alive tasks executed against the server connected to.
- Implemented dynamic type conversion in OpcValue.As<T>()
- Added new AsValue<T>() method to OpcValue to provide a generic definition of the value using the new OpcValue<T> class.
- OpcNamespace now provides the namespace string / uri using the new OpcNamespace.Value property.
- Default transport limits increased DefaultMaxByteStringLength from 64 KB to 5 MB and DefaultMaxMessageSize from 1 MB to 10 MB.
- Solved inline handled Exceptions in XmlSerializer during startup.
- Improved Exception handling and tracing during data encoding and decoding (see OpcTypes.EncodingFailed and OpcTypes.DecodingFailed).
- Introduced decoding and encoding related tracing support using the new OpcEncodingStackFrame and OpcEncodingStackTrace classes.
- Added “int ResolveIndex(string namespaceValue)” and “string ResolveValue(int namespaceIndex)” to IOpcNamespaceResolver.
- OpcClient now provides service methods to simplify calls to CallMethod using the new generic methods CallAction and CallFunc.
- Added additional overloads to OpcBrowseNode to support OpcNodeIds to refer to specific types of references.
- OpcNodeInfo now provides additional overloads of Children(…) and Parents(…) accepting a OpcReferenceType value.
- OpcResult now provides with GetBaseResult() the root result.
- OpcEventNode.EventTypeId is now writable using an added setter.
- Introduced OpcEventTypeAttribute to define client-side declared custom event types used to handle event notification data in client applications.
- Corrected wrong used status code while decoding in foundation stack (before: BadEncodingError, now: BadDecodingError).
- Added missing SourceType property to Opc.Ua.Schema.Binary.FieldType.
- Added missing BaseType property to Opc.Ua.Schema.Binary.BaseType.

FIXED

- Wrong used variable when configuring Opc.UaFx.Client.OpcSubscription.LifetimeCount.
- Issues during serialization of OpcException objects.

CHANGED

- OpcDataTypeAttribute is now applicable on classes and structures.
- Unified OpcNodeManager.NamespaceIndexes and OpcNodeManager.NamespaceUris into new OpcNodeManager.Namespaces.
- Renamed IOpcReadOnlyDataStore to IOpcReadOnlyNodeDataStore.
- IOpcNamespaceResolver “Uri Resolve(int namespaceIndex)” and “int Resolve(Uri namespaceUri)” are now obsolete.

2.7.5.1 (released 2019-08-15)

NEW

- Added use of OpcNominalNodeIdFactory (via OpcNodeId.Factory) to resolve OPC Classic tag names using the OPC Classic Server specific tag separator.

FIXED

- Issue with multiple times serialized exception information which causes an SerializationException instead of the exception itself.

2.7.5.0 (released 2019-08-13)

NEW

- Improved API documentation: IOpcNode, IOpcNodeInfo and OpcNode.
- Stack change to assure that a node-manager lock is hold as long monitored items are created, modified or deleted.
- Updated to System.ServiceModel.Primitives 4.5.3, because in System.Private.ServiceModel 4.5.x a bug was fixed that the “System.Private.ServiceModel.dll” was not copied for .NET Core applications when publishing for Windows (win-xxx).

FIXED

- Issues with compiling using .NET Native tool chain.
- Issues with multiplied subscription notifications and events after reconnecting to the server.
- Issues with inline parsed OpcNodeId's when creating one using a string identifier.
- Issue with wrong used base address and security policies in OpcDiscoveryServer.
- Issue with re-created OpcMonitoredItem instances which additionally resulted into the issue that comparing OpcMonitoredItem instances not longer worked.

CHANGED

- OpcMonitoredItem.AutoFlushCache has been replaced with new OpcMonitoredItem.QueueMode property to better express the mechanism used when the notification data item queue of an monitored item overflows.

2.7.4.0 (released 2019-06-07)

NEW

- Domain name resolution used while validating client / server certificates now uses fallback mechanisms in case there is no DNS server configured in the operating system.
- Re-implemented explicit use of OperationTimeout when using HTTPS (which is not longer part in the foundation stack).
- Implemented MonitoredItemsCreated, MonitoredItemsModified and MonitoredItemsDeleted events in

OpcNodeManager.

FIXED

- SocketExceptions on macOS while validating certificate domains using domain name resolution although there is no (available) DNS server configured in the operating system.

CHANGED

- Some int-based properties in OpcMonitoredItem, OpcServiceCounter, OpcSessionDiagnostics and OpcServerDiagnostics to long as their property type to assure CLS compliance while accessing the unsigned API of the foundation stack.

2.7.3.1 (released 2019-05-23)

NEW

- [Client SDK] Added support for target framework .NET 4.6.

FIXED

- Issues with connecting to a server using https on macOS and Linux, which results into the OpcException: An unrecognized response was received from the server.

2.7.3.0 (released 2019-05-17)

NEW

- Replaced local X509Certificate2 instance activation by using OpcCertificateManager.QueryInstance to load certificate information. This enables custom X509Certificate2 creation using an event handler.
- Introduced new OpcCertificateStores.PathProbings property to enable custom probing mechanism to determine the certificate store paths to use by default.
- Improved user experience regarding different (smaller) issues using subscriptions and monitored items (especially in automatically reconnecting clients).
- Introduced additional AddMonitoredItem(…) overloads on OpcSubscription.

FIXED

- Issue with connecting to a servers using https, which results into the OpcException: An unrecognized response was received from the server.

CHANGED

- Reviewed implementation of OpcMonitoredItem and OpcSubscription: Changed some “int”-based properties to use “long” as the property type to support underlaying “uint” properties used by the foundations stack to keep CLSCompliance.

2.7.2.0 (released 2019-05-10)

NEW

- Support of TypeDefinition information when browsing intance nodes through new OpcInstanceNodeInfo class.
- Added Socket property to the OpcSession class to identify the socket identity information for the used local and remote endpoints.

- Changed product icon of NuGet package.
- [Client SDK] Enabled support for IL2CPP in Unity projects. This addresses some Overflow exceptions reported while IL2CPP has been executed.

FIXED

- Issues when browsing more than one reference type at once which resultet into an ArgumentOutOfRangeException.
- Issues with sometimes outdated subscription and monitored item information when the client automatically reconnects to the server after the connection got lost.
- Issue with sometimes missing Client and Subscription based notification events - e.g. NotificationReceived event has been missed on OpcClient.

2.7.1.0 (released 2019-03-25)

NEW

- Implemented rejection of requested node identifiers using unknown namespace index values (in AddNodes service, BadNodeIdInvalid).
- Implemented use of node (identifier) specific node manager to assure that a node being added is not added to the nodes of the node manager of the parent node in case there the new node refers to the namespace maintained by a different node manager.
- Node access and write flags are now applied recursively when adding nodes by a client.

FIXED

- Issue with bad node rejected in case of a client defined node identifier using AddNodes.
- Issues with custom node attributes used to setup the write and access attributes of nodes added by clients.
- Corrected not nicely formed namespace uri of default node manager (removed unnecessary '/').
- Issue with wrong wrapped .NET array types using OpcValue instances.
- Issue with wrong wrapped UA FX array types using OpcValue instances.
- Issue with missing DataType information in case there a OpcVariableNode (non generic) is initialized with an initial value using its constructor followed by a change of its DataType property to the appropriate type. Result: The DataType property value was not accepted.

2.7.0.2 (released 2019-03-15)

- **FIXED:** Wrong used addressing of unmanaged memory (using OPC Classic) which especially can fail in processes using large address awarness (see LARGEADDRESSAWARE).

2.7.0.1 (released 2019-03-15)

- **FIXED:** Issue with locales which are supported by the OPC Classic servers but not by the operating system of the client.
- **FIXED:** Issue with addressing OPC Classic servers running on a 64 bit system.

2.7.0.0 (released 2019-03-14)

- **NEW:** Completly reworked AddNode support in OpcClient including a whole new set of OpcAddNode command types like: OpcAddFolderNode, OpcAddObjectNode, OpcAddDataVariableNode,

OpcAddAnalogItemNode and some more.
- **NEW:** First time full support of AddNode service calls in OpcServer accepting every supported type definition to add a new node into the address space.
- **NEW:** Completly reworked DeleteNode, AddReference and DeleteReference support in OpcClient including a whole new definition of the OpcDeleteNode, OpcAddReference and OpcDeleteReference command.
- **NEW:** Reworked support of DeleteNode, AddReference and DeleteReference service calls in OpcServer using additional request validations.
- **NEW:** Introduced new OpcAddNodeResult class which provides a specialization of the OpcResult class to provide the used NodeId and the command instance from which the result originates.
- **NEW:** OpcNodeManager now provides virtual methods called when adding or deleting a node to / from the address space of the manager.
- **NEW:** Improved exception details when using the services AddNodes, DeleteNodes, AddReferences and DeleteReferences.
- **NEW:** OpcServerIdentity constructor overloads accepts a string value as password.
- **NEW:** OpcServer events: RequestProcessing, RequestValidating, RequestValidated and RequestProcessed to pre- and post process client requests and the send responses.
- **NEW:** OpcException always provides Source, StackTrace, Data and HelpLink information were available and not does not longer skip them in case of a different source exception.
- **NEW:** IOpcNode.Child(...) method to retrieve an IOpcNode child instance using the name (= browse name) of the child node.
- **NEW:** Additional OpcServerIdentity.ChangePassword(string) method overloads.
- **NEW:** Additional OpcUserNameAcl.AddEntry(...) method overloads to define a custom OpcServerIdentity.
- **NEW:** Enabled subclassing of OpcResult using custom types.
- **CHANGED:** Removed OpcServerIdentity.Password property, because of there is no need to publish the internal stored password information and it is not a good practice to provide references to an array through properties.
- **CHANGED:** Removed OpcTransformUserPasswordDelegate delegate.
- **CHANGED:** Replaced OpcUserNameAcl.PasswordTransform with virtual Matches and TransformPassword method in OpcServerIdentity.
- **FIXED:** Issue when writing built in structured values like OpcEngineeringUnitInfo or OpcValueRange using OpcClient.WriteNode.
- **FIXED:** Issue when disconnecting / stopping the client / server which lead to a crash in case there was still at least one connect attemption pending.
- **FIXED:** Issue with StackOverflowException when changing the OpcServer.Configuration instance.

2.6.0.0 (released 2019-02-20)

- **NEW:** Implemented OPC Classic support in OpcClient just using an address like "opc.com://<host>(:<port>)/<progId>/<classId>". (only in .NET Framework)
- **NEW:** Implemented new OpcClassicInterface, OpcClassicInterfaces and OpcClassicInterfaceCollection classes and OpcClassicInterfaceCategory enumeration. (only in .NET Framework)
- **NEW:** Implemented new OpcClassicApplicationDescription, OpcClassicApplicationDescriptionCollection, OpcClassicEndpointDescription and OpcClassicEndpointDescriptionCollection classes. (only in .NET Framework)
- **NEW:** Implemented OPC Classic discovery using the new OpcClassicDiscoveryClient class which uses “OpcEnum.exe” (a COM application provided by the OPC Classic runtime environment). (only in

.NET Framework)

- **NEW:** Enhanced existing OpcDataChangeFilter to support the configuration of the Trigger, DeadbandType and DeadbandValue. Using the Trigger property it is possible to monitor items using the SourceTimestamp as well.
- **NEW:** Implemented additional OpcClient.SubscribeDataChange overloads and additional OpcDataSubscribeDataChange constructor overloads accepting a value of the OpcDataChangeTrigger enumeration.
- **NEW:** An predefined ApplicationName is now used as the SubjectName of the certificate when (automatically) creating a new certificate used by client / server applications.
- **NEW:** Implemented generic OpcApplicationInstance class implementing IOpcApplicationInstance.
- **NEW:** OpcClient and OpcServerBase use the OpcApplicationInstance class as their base class.
- **NEW:** Implemented generic OpcSecurity class.
- **NEW:** OpcClient derives from OpcApplicationInstance class.
- **NEW:** OpcServer derives from OpcApplicationInstance class.
- **NEW:** Centralized application configuration of client / server application instances using OpcApplicationInstance.
- **NEW:** Implemented generic OpcTransport configuration class and its use in IOpcApplicationInstance.
- **NEW:** Implemented generic OpcSecurity configuration class and its use in IOpcApplicationInstance.
- **NEW:** Added Security and Transport properties to IOpcApplicationInstance.
- **NEW:** Added CertificateValidationFailed event to IOpcApplicationInstance.
- **NEW:** OpcSecurityPolicyCollection now implementes a public constructor.
- **CHANGED:** Removed OpcClientTransport class (replaced by new OpcTransport class).
- **CHANGED:** Removed OpcServerTransport class (replaced by new OpcTransport class).
- **CHANGED:** The default server information provided by OpcServerBase now prefers a predefined ApplicationName.
- **CHANGED:** OpcSecurityPolicyCollection does not longer provide a constructor with a OpcServerBase instance. The according Owner property has been removed as well.

2.5.7.0 (released 2019-02-06)

- **NEW:** OpcClient.WriteNode now directly accepts null as the value without the need of a new OpcValue instance.
- **NEW:** The OpcServer class provides new ctor overloads which use the default address “opc.tcp” + “localhost” + “4840” (the default port).
- **NEW:** The whole node tree can now created inline using new ctors of OpcObjectNode and OpcFolderNode to specify their children directly when they are created.
- **NEW:** Ensured that read/write operations on attributes are cancelled in case there a custom read/write callback returns a “bad” result. In case of a “good” result the processing of the read/write operation continues.
- **NEW:** API its signature only accepts “nodeId:string” to identify a node now parses inline the specified string value to ensure a more fluent usage of the framework API.
- **CHANGED:** The application certificate is not longer stored in .\CertificateStores\Trusted, now it is stored (by default) in .\CertificateStores\App to improve startup performance when searching for the application certificate.
- **CHANGED:** Changed use of LinkTime to LastWriteTime of assembly when determining OpcServerInfo from an assembly.
- **FIXED:** Issue with not applied initial value when using OpcDataVariableNode.ctor(parent, name, value).
- **FIXED:** Included missed value checks (like BadTypeMismatch) in cases there the Value attribute is

written.

2.5.6.0 (released 2018-10-19 [Client SDK], 2019-02-06 [Client+Server SDK])

- **NEW:** Added enhanced Android linking support when linking SDK+User for additional APIs.
- **NEW:** Enhanced default certificate store directory to first try to use the local directory, then CommonApplicationData and then ApplicationData.
- **NEW:** Additional support of .NET Framework 4.7.1 as the target framework to not longer refer to 12 additional system assemblies caused by .NET Standard references in .NET Framework 4.7.1. See https://github.com/Microsoft/dotnet/blob/master/releases/net471/KnownIssues/514195-Targeting%20.NET%20Framework%204.7.1%20copies%20extra%20files%20to%20your%20bin%20directory.md
- **FIXED:** Issue when checking if a certificate store already contains a certificate. The store always returned true.
- **FIXED:** Solved issues when browsing in Android SDK+User linked environment.

2.5.5.0 (released 2018-10-11 [Client SDK], 2019-02-06 [Client+Server SDK])

- **NEW:** Extracted API for exclusive Client development from OPC UA Advanced.
- **NEW:** CertificateRequested event on OpcClient to query an alternative certificate in case there a existing application instance certificate could not found or can't be used. This event can also used to determine the reason why a new certificate is required. This additionally enables the possibility to notify / query the user of the client about this need.
- **NEW:** In cases there ".\CertificateStores\" (the default directory for certificates) is not writable the path "$(AppData)\OPC Foundation\CertificateStores\" is instead used as the default directory to maintain certificates.
- **NEW:** Increased default lifetime of a certificate from 12 months (= 1 year) to 600 months (= 50 years).
- **NEW:** Reduced default key size from 2048 to 1024 to support older certificates.
- **NEW:** Less assembly size and less dependencies through reorganization of the internal licensing mechanism for .NET Standard.
- **NEW:** In case of failed socket binding an exception is not longer caught nor thrown as a generic exception - instead, it is forwarded to the caller.
- **NEW:** Binding property on OpcServer which provides a instance of the new OpcServerBinding class.
- **NEW:** OpcServerBinding (OpcServerBase.Binding property) provides the possibility to define the address scope the server binds to. By default it will bind (as before) to any IPv4 and to any IPv6 address using the specified port number.
- **NEW:** The low level Read / Write operations of node managers now provide OpcNodeAccessTokens with operation result information.
- **NEW:** Added documentation for OpcSubscription, OpcMonitoredItem, OpcMonitoredItemEventArgs, OpcMonitoredItemEventHandler and much some regarding subscriptions.
- **CHANGED:** Default MinimumCertificateKeySize from 2048 to 1024 to support lower key sizes used by older certificates.
- **CHANGED:** When changing the application certificate of a client or server instance using the Certificate property the store path type of the application certificate store changes to System. To use the store path type Directory you have to change the store path type manually and have to ensure that certificate can be found in the application certificate store - after changing the Certificate property.
- **FIXED:** OpcServer.Security.AutoAcceptUntrustedCertificates when using secured endpoint policies did not auto accept untrusted certificates.

- **FIXED:** Problem when dynamically determining start time of node history to read count-based.
- **FIXED:** Issue with rejected path of certificate application store when manually changing the application certificate.
- **FIXED:** Issue with so far not comparable monitored item and session instances.
- **FIXED:** Issue when writing a node value using OpcValue encapsulated raw values.
- **FIXED:** Wrong use of AutoAcceptUntrustedCertificates in server application instances.
- **FIXED:** Wrong used start time when reading historical values without explictly specifying a start time.

2.5.4.0 (released 2018-08-27)

- **NEW:** Support for automatic history-creation in NodeHistorian through client written values (AutoUpdateHistory property).
- **NEW:** OpcNodeHistorian differs node-changes and only transfers value-cahnges to the history.
- **NEW:** Improved code documentation of the servers Subscription / MonitoredItem API.
- **NEW:** Server.OpcSubscription and Server.OpcMonitoredItem instances are now comparable.
- **FIXED** OpcVariableNode.IsHistorizing is not longer reset after calling OpcNodeManager.CreateNodes.
- **FIXED:** Issue when writing to variable nodes using an undefined DataType attribute (did not work and led to an exception).
- **CHANGED:** OpcNode.BeforeApplyChanges and OpcNode.AfterApplyChanges are only raised in cases there any change exists (see OpcNode.HasPendingChanges).

2.5.3.0 (released 2018-08-21)

- **NEW:** Implemented automatic value arrangement after reading a node value (server side) using clients requested encoding and value range.
- **NEW:** New OpcClientTransport class accessible via OpcClient.Transport to setup transport options like timeouts and limits.
- **NEW:** New OpcServerTransport class accessible via OpcServer.Transport to setup transport options like timeouts and limits.
- **FIXED:** Issue with wrong used certificate data source when validating a X509 identity token.

2.5.2.5 (released 2018-08-07)

- **NEW:** Removed NuGet dependencies enforced (in .NET Standard) to reference in user projects.
- **NEW:** Simplified content of *.deps.json (in .NET Standard).

2.5.2.4 (released 2018-07-31)

- **FIXED:** Issue with reporting global events using OpcServer and OpcNodeManager instances.
- **NEW:** Implemented support of Null-OpcName instances then using OpcNominalNodeIdFactory.
- **NEW:** OpcContext now always provides an appropriate OpcNodeIdFactory instance, at least the global instance defined by OpcNodeId.Factory.

2.5.2.3 (released 2018-07-23)

- **NEW:** NuGet Updated.

2.5.2.2 (released 2018-07-23)

- **FIXED:** InvalidCastException when the assembly was loaded with Assembly.LoadFrom().

2.5.2.1 (released 2018-07-13)

- **FIXED:** DllNotFoundException when trying to use specific types before creating an OpcClient or OpcServer.

2.5.2.0 (released 2018-07-06)

- **NEW:** .NET Standard / .NET Core Licensing and Support has been released.

2.5.1.1 (released 2018-07-03)

- **NEW:** Demo license has been renewed to provide a new license period.

2.5.1.0 (released 2018-06-15)

- **NEW:** MimeType-Support in OpcFileNode and OpcFileInfo.
- **NEW:** Improved performance when accessing OpcContext references in OpcNodes (internally).
- **NEW:** Improved performance / memory usage when accessing the children of a OpcNode (internally).
- **NEW:** Removed inline and by default always prepared history configuration nodes for all variable nodes. Instead it is now created on-demand.
- **NEW:** Removed internal cross-references and use of heavy OnXy callbacks defined by the foundation. Instead we use now a provider pattern.
- **NEW:** Introduced IOpcNodeInfo to not longer create temporary OpcNode instances. This improves performance and memory usage.
- **NEW:** IOpcNodeInfo which is used in scenarios there a reference to the origin node is not required. This also reduces the amount of memory required to process internal node operations / evaluations.
- **NEW:** Introduced cache for custom data types to not longer repeatedly reflect their attribute to improve performance.
- **NEW:** Exceptions which are thrown during the execution of CreateNodes(...) are now passed to the source there OpcServer.Start has been called.
- **CHANGED:** IsNodeAccessible(..., IOpcNode node) to IsNodeAccessible(..., IOpcNodeInfo node)

2.5.0.3 (released 2018-05-24)

- **FIXED:** Issue with not working external references when using multiple node managers which share the same custom node as the root of the nodes provided.

2.5.0.2 (released 2018-05-23)

- **FIXED:** Issue with missing nodes when using multiple node managers referencing the same system node.

2.5.0.1 (released 2018-05-16)

- **FIXED:** NullReferenceException in OpcMethodNode when using a predefined node as the parent of the method node.
- **FIXED:** Not longer displayed OpcDataVariableNode in servers address space, when using the ctor(parent : IOpcNode, name : OpcName).
- **CHANGED:** Downgraded sample projects to C# language version 6.0 to ensure that the samples will compile in older versions of Visual Studio.

2.5.0.0 (released 2018-05-10)

- **NEW:** Added support for creating nodes using a custom node identifier.
- **NEW:** Streamlined constructor overloads in all node classes.
- **NEW:** Added much so far missed code documentation on node classes.
- **NEW:** Implemented for each property of a node a according XyNode property which provides the reference to the property-node that represents the value of the so far available Xy properties.
- **NEW:** Implemented support for lazy initialiation of nodes which allows to define the Xy property-values to be set-up while they are applied to their property-nodes on server-startup.
- **NEW:** Completely reviewed implementation of OpcNodeId and OpcNamespace. Partially re-implemented both classes to ensure a maximum of compatibility to foundation defined node identifier standards including the features provided by this framework.
- **NEW:** The OpcNodeId now supports identifiers using byte-arrays, Guid's and unsigned integers.
- **NEW:** The OpcNodeId now expresses the type of identifier value in its string representations generated using OpcNodeId.ToString.
- **NEW:** The OpcNodeId now supports UTF8 encoded special characters in an url-formatted node identifier.
- **NEW:** The OpcNodeId now support parsing a foundation formatted node identifiers.
- **NEW:** The OpcNodeId now provides additional properties like OriginalString and OriginalFormat.
- **NEW:** Reworked OpcName implementation to not longer combine the BrowseName, DisplayName and SymbolicName into one instance / class.
- **NEW:** IOpcNode, OpcNode and OpcNodeInfo now provide direct properties for the DisplayName (and SymbolicName) of a node.
- **NEW:** OpcNodeInfo now supports the Children and Parent browsing using a combination of the OpcNodeCategory enum members.
- **NEW:** OpcClient.SubscribeEventRaise methods and OpcSubscribeEventRaise now support an OpcEventFilter instance for event filtering.
- **NEW:** Reviewed existing API regarding UA Alarm + Events and adjusted it for a more fluent / intuitive interface including some changes to use lazy initialization to improve performance.
- **NEW:** Reworked OpcEventRaiseEventArgs to define the Event property for EventNode snapshot information using event raw data.
- **NEW:** Added MonitoredItem property to OpcDataChangeEventArgs and OpcEventRaiseEventArgs.
- **NEW:** Added events DataChangedReceived and EventRaiseReceived to OpcSubscription.
- **NEW:** Implemented OpcEventType enumeration and OpcEventTypes to simplify access to the different event types supported.
- **NEW:** Implemented different state machine and state / transition variable nodes.
- **NEW:** Implemented DialogRequested event in OpcClient to provide a simple way to process OpcDialogConditionNode instances.
- **NEW:** Implemented support for foundation specified event types. This are: OpcConditionNode, OpcAcknowledgeableConditionNode, OpcDialogConditionNode, OpcAlarmConditionNode,

OpcDiscreteAlarmNode, OpcLimitAlarmNode, OpcOffNormalAlarmNode, OpcSystemOffNormalAlarmNode, OpcTripAlarmNode, OpcExclusiveLimitAlarmNode, OpcNonExclusiveLimitAlarmNode, OpcExclusiveDeviationAlarmNode, OpcExclusiveLevelAlarmNode, OpcExclusiveRateOfChangeAlarmNode, OpcNonExclusiveDeviationAlarmNode, OpcNonExclusiveLevelAlarmNode and OpcNonExclusiveRateOfChangeAlarmNode.

- **FIXED:** Wrong name of method GetOutpuArguments to GetOutputArguments in OpcMethodNodeInfo class.
- **FIXED:** Issue with DBNull values used in case there is no default value for an argument specified in method nodes.
- **CHANGED:** Corrected misspelled members.
- **CHANGED:** Streamlined API through added additional descriptive parts to class and delegates names. In general changed the naming of Data to DataSet, DataItem to DataSetItem and EventRaise to Event.
- **CHANGED:** OpcNodeId.ToString formats now by default using the OpcNodeIdFormat.Foundation format.
- **CHANGED:** Order of subscription events from MonitoredItem → Client → Subscription to MonitoredItem → Subscription → Client. To ensure a more logic event order using bubbling.
- **CHANGED:** Renamed OpcMonitoredItem.ReceivedDataChange to OpcMonitoredItem.DataChangeReceived.
- **CHANGED:** Renamed OpcMonitoredItem.ReceivedEventRaise to OpcMonitoredItem.EventRaiseReceived.

BREAKING CHANGES (through renaming):

- OpcSubscribeEventRaise → OpcSubscribeEvent
- OpcDataChangeEventArgs → OpcDataChangeReceivedEventArgs
- OpcDataChangeEventHandler → OpcDataChangeReceivedEventHandler
- OpcEventRaiseEventArgs → OpcEventReceivedEventArgs
- OpcEventRaiseEventHandler → OpcEventReceivedEventHandler
- OpcNotificationEventArgs → OpcNotificationReceivedEventArgs
- OpcNotificationEventHandler → OpcNotificationReceivedEventHandler
- OpcDataChangeCallback → OpcDataChangeReceivedCallback
- OpcEventRaiseCallback → OpcEventReceivedCallback
- IOpcNotificationData → IOpcNotificationDataSet
- OpcNotificationData → OpcNotificationDataSet
- OpcNotificationDataCollection → OpcNotificationDataSetCollection
- OpcNotificationDataReadOnlyCollection → OpcNotificationDataSetReadOnlyCollection
- OpcNotificationDataItem → OpcNotificationDataSetItem
- OpcNotificationDataItemCollection → OpcNotificationDataSetItemCollection
- OpcNotificationDataItemReadOnlyCollection → OpcNotificationDataSetItemReadOnlyCollection
- OpcDataChange → OpcDataChangeSet
- OpcDataChangeItem → OpcDataChangeSetItem
- OpcEventRaise → OpcEventDataSet
- OpcEventRaiseItem → OpcEventDataSetItem
- OpcMonitoredItem
  - .EventRaiseReceived → EventReceived
  - .LastEventRaise → LastEvent
  - .OnEventRaiseReceived → OnEventReceived
- OpcSubscription
  - .EventRaiseReceived → EventReceived

  - .ReceivedEventRaiseCallback → ReceivedEventCallback
  - .OnEventRaiseReceived → OnEventReceived
- OpcClient
  - .EventRaiseReceived → EventReceived
  - .OnEventRaiseReceived → OnEventReceived
  - .SubscribeEventRaise → SubscribeEvent

2.3.2.0 (released 2018-03-13)

- **FIXED:** Issue with wrong use of DebuggerDisplayAttribute in OpcVariableNode and OpcNodeInfo.
- **FIXED:** Issue with reading / writing custom DataType attribute values on OpcVariableNodes.
- **CHANGED:** OpcVariableNode to define an abstract base class.
- **CHANGED:** Removed multiple local foundation node references to reduce memory usage.
- **NEW:** Enhanced implementation of OpcVariableNode especially for enumeration types.
- **NEW:** Implemented new nodes OpcTypeNode and OpcDataTypeNode. The OpcDataTypeNode does support the definition of custom enumeration types.
- **NEW:** Implemented enhanced support to simplify browsing of enumerated data types. Using the new OpcTypeNodeInfo.GetEnumMembers() method provides all the information available on an enumerated type (also see the new OpcTypeNodeInfo.IsEnum property).

2.3.1.1 (released 2018-02-26)

- **FIXED:** Issue with loading the OpcApplicationConfiguration from a file, because of a required DataContract attribute on the OpcApplicationConfiguration class which was not required in older framework versions.
- **NEW:** Implemented additional ctor overloads in OpcEngineeringUnitInfo to simplify initialization of a specific unit.
- **NEW:** Implemented ToString overloads in OpcEngineeringUnitInfo and OpcValueRange.
- **NEW:** CommonCode property in OpcEngineeringUnitInfo + static GetCommonCode(int unitId) method.
- **NEW:** DefaultNamespaceUri in OpcEngineeringUnitInfo when using a constructor without an explicit namespace parameter. This is not the case using the default constructor.

2.3.1.0 (released 2018-02-22)

- **ATTENTION:** Starting with this version a reference to the NuGet package Portable.BouncyCastle (at least V1.8.1.3) or BouncyCastle.Crypto assembly (at least V1.8.1.146) is required.
- **ATTENTION:** Using a server address with the scheme 'https' requires references to the NuGet packages 'Microsoft.AspNetCore.Server.Kestrel (1.1.3)' and 'Microsoft.AspNetCore.Server.Kestrel.Https (1.1.3)'.
- **NEW:** Implemented the Range DataType as OpcValueRange.
- **NEW:** Implemented the EUInformation DataType as OpcEngineeringUnitInfo.
- **NEW:** Implemented new node type AnlogItemType as OpcAnalogItemNode.
- **NEW:** Support of DataValues using the data types Range and EUInformation.
- **NEW:** Advanced support for browsing nodes using the AnalogItemType-Definition thorugh the new OpcAnalogItemNodeInfo.
- **NEW:** Updated to target framework .NET Framework 4.6.

2.3.0.0 (released 2018-02-13)

- **NEW:** Updated to foundation stack V1.03.351.0.
- **NEW:** Implemented new Security option for client / server to define additional certificate validation rules (see OpcClient/OpcServer.Security.CertificateValidationRules).
- **CHANGED:** The framework does not longer support Windows based authentication using WinLogon-API for OPC UA server authentication.
- **CHANGED:** Application certificates are now created using SHA256 (the default defined by the foundation). This does not influence any running code. This change will only have an effect when a new certificate is generated.

2.2.2.0 (released 2017-12-11)

- **NEW:** Enhanced filter construction using operator overloads to get a more natural API.
- **NEW:** Introduced OpcText to represent localized human readable string values.
- **FIXED:** Issue with missing use of configured operation timeout using OpcClient.Connect() while discovering the server endpoints.
- **CHANGED:** OpcValue → Simplified ToString implementation to just provide “null” for null references or the Value.ToString().

2.2.1.0 (released 2017-09-06)

- **NEW:** Added support for incomplete certificate chains in cases when the opponent provides a certificate which is signed by an unknown issuer. This is for example the case when trying to connect to a Siemens SIMOTION.
- **NEW:** Enhanced support for failed certificate validations using the CertificateValidationFailed events in the OpcClient and OpcServer class.

2.2.0.0 (released 2017-09-01)

- **NEW:** Added the GetReferenceType to resolve the according OpcReferenceType for a given OpcNodeId of the type of reference.
- **CHANGED:** Removed special handling of the attribute DataType in the OpcNodeInfo.Attribute(). This means that there is no longer an OpcDataType value provided. Instead, the data type dependent OpcNodeId is provided.
- **NEW:** Implemented basic support for object Nodes and extended support for variable Nodes, also implemented generic support for type Nodes (including ObjectTypes, VariableTypes and DataTypes) while browsing Nodes.
- **NEW:** Using the OpcVariableNodeInfo, OpcObjectNodeInfo and OpcTypeNodeInfo, it is now possible to browse the address space using additional information about the Nodes provided. This also includes e.g. the type information of a variable Node including all its type specific metadata including super- und subtypes.
- **NEW:** Prepared the framework for use under .NET Standard and .NET Core 2.0.
- **NEW:** Introduced the LicenseInfo property on all Licenser classes to provide some information about the license conditions used.

2.1.0.0 (released 2017-08-01)

- **NEW:** Updated to OPC UA Foundation stack V1.3.342.0.

2.0.1.1 (released 2017-06-05)

- **NEW:** RegisterAddress / UnregisterAddress methods and new Addresses method to maintain the different base addresses of an OpcServerBase instance.
- **FIXED:** Wrong naming of IsNodeAccessable in OpcNodeManager to IsNodeAccessible.
- **FIXED:** Issue with malformed Node namespaces using a namespace URI just built from a scheme in case the OpcNamespace.ToString() is called (= FormatException).
- **FIXED:** Issue with wrong implicit interpretation of status codes to result information.

2.0.1.0 (released 2017-05-24)

- **NEW:** Added much more details to the message of an OpcException and to the description of an OpcStatus / OpcResult instance to better indicate the source / reason / outcome of an operation.

2.0.0.0 (released 2017-05-21)

- **NEW:** Changed the Assembly Version to 2.0.0.0.
- **NEW:** Implemented OpcCertificateSettings to configure an application instance certificate which can be created through the OpcCertificateManager.
- **FIXED:** Issue with wrongly configured application certificates using a store path in cases when the certificate can not be located under such a path.
- **FIXED:** Issue with wrongly stored username-password pair in the UserName ACL that led to unsuccessful user authentication.
- **FIXED:** Issue when creating an OpcNodeId using the string+int32 pair, where the int32 was not used for the namespace index of the new Node identifier.
- **FIXED:** Issue with OpcServerBase.Certificate and OpcClient.Certificate in the case that the certificate is not stored in the ApplicationStore of the certificate stores.
- **NEW:** Introduced OpcClientCertificateStores (available through OpcClient.CertificateStores) and OpcServerCertitificateStores (available through OpcServerBase.CertificateStores) to provide a fluent API for Client and Server to manage and maintain their certificate stores.
- **NEW:** Introduced OpcClientServices (available through OpcClient.Services) to provide a service based API to set up the different services used by the Client.
- **REMOVED:** “OpcValue.SourceLabel” and “OpcValue.ServerLabel”. This also includes that the “OpcValueLabel” class does no longer exist. The provided properties are now provided directly via the “OpcValue” class. Just make use of the “OpcValue.ServerXy” and “OpcValue.SourceXy” properties.
- **NEW:** IOpcWriteNodesService does now support an automatic discovery of which value the Server does expect for the SourceTimestamp of an OpcValue. The automatic determination can be disabled by setting an explicit OpcTimestampSource using the IOpcWriteNodesService.TimestampSource property. Note: The IOpcWriteNodesService instance can be accessed using the OpcClient.Services.WriteNodes property.
- **NEW:** Removed usage of the obsolete ResolveNodeId method and implemented the new Resolve method on the OpcNodeId instead to determine the fully qualified Node identifier using the Server's namespaces array on demand. This does now support the Node access using the fully qualified Node identifier without the need to know the index of the dedicated Node namespace.
- **NEW:** Implemented an additional overload of the UpdateCertificate method to manually define whether a certificate is to be created in case there is a certificate missing or not valid. So far this

was only possible by using the static AutoCreateCertificate property.
- **NEW:** Implemented the new properties OpcClient.Certificate (to directly the define a custom Client certificate) and OpcClient.Security (to set up Client security options within one class).
- **CHANGED:** Renamed property “OpcSecurity.Policies” to “OpcSecurity.EndpointPolicies” to match the “OpcClientSecurity.EndpointPolicy” model.
- **CHANGED:** Defined the default value of “OpcCertificateManager.AutoCreateCertificate” as true.
- **NEW:** Implemented the new properties in Opc.UaFx.Server.OpcSecurity: AutoAcceptUntrustedCertificates and AutoCreateCertificate.
- **NEW:** Implemented the new OpcServer.Certificate property to match the same API model as the OpcClient implements.
- **CHANGED:** Behavior used to select an endpoint to connect to in order to no longer to be negotiable in case there is an explicit endpoint policy defined.
- **CHANGED:** Default of AutoAcceptUntrustedCertificates in SecurityConfiguration to true.
- **NEW:** Implemented support for a DefaultNodeManager - this is either the first manager in the list of managers defined or the internally defined default Node manager which defines the namespace as “<serveraddress>/nodes/”. There is now also the possibility to define just a callback method to provide the Nodes for the default Node manager's address space.
- **MOVED:** “OpcClient.PreferredPolicy” property to “OpcClient.Security.EndpointPolicy”.
- **MOVED:** “OpcClient.UseDomainChecks” property to “OpcClient.Security.VerifyServersCertificateDomains” and changed its default value to false.
- **MOVED:** “OpcClient.UserIdentity” property to “OpcClient.Security.UserIdentity” and changed its type from the foundation type IUserIdentity to OpcUserIdentity.
- **MOVED:** “OpcClient.UseOnlySecureEndpoints” property to “OpcClient.Security.UseOnlySecureEndpoints” and changed its default value to false.
- **REMOVED:** Obsolete “OpcClient.UseAutoNodeIdResolution” property.
- **NEW:** Reworked / reviewed OpcIdentity derivates and defined the OpcCertificateIdentity, OpcWindowsIdentity, OpcServerIdentity and OpcClientIdentity as new derivates of OpcUserIdentity (so far they just used the OpcIdentity as their base class). Also changed their construction behavior to simplify the process of identity creation on Client and Server side.
- **FIXED:** OpcRequestType.Call has been defined as OpcRequestType.Cancel so far. This lead to a nonfunctional use of the OpcRequestType.Call, because it has then been used like the OpcRequestType.Cancel.
- **REMOVED:** OpcCertificateManager.AutoCreateCertificate property, because it is now used per instance using OpcClient.Security.AutoCreateCertificate or OpcServer.Security.AutoCreateCertificate. Also these properties are by default set to the value true.
- **REMOVED:** OpcServer.Security.Owner property to reduce redundant reference information and simplify the API.
- **RENAMED:** “Server.OpcSecurity” to “Server.OpcServerSecurity”.
- **RENAMED:** “OpcServerSecurity.Anonymous” to “OpcServerSecurity.AnonymousAcl”
- **RENAMED:** “OpcServerSecurity.Certificate” to “OpcServerSecurity.CertificateAcl”
- **RENAMED:** “OpcServerSecurity.IssuedToken” to “OpcServerSecurity.IssuedTokenAcl”
- **RENAMED:** “OpcServerSecurity.UserName” to “OpcServerSecurity.UserNameAcl”
- **RENAMED:** “OpcClient.PreferredLocales” to “OpcClient.Locales”
- **RENAMED:** “OpcClient.ReceivedNotification” to “OpcClient.NotificationReceived”
- **RENAMED:** “OpcClient.OnReceivedNotification” to “OpcClient.OnNotificationReceived”
- **RENAMED:** “OpcSubscription.ReceivedNotification” to “OpcSubscription.NotificationReceived”
- **RENAMED:** “OpcSubscription.OnReceivedNotification” to “OpcSubscription.OnNotificationReceived”
- **MOVED:** OpcNodeCommand from Opc.UaFx.Client to Opc.UaFx.Services.
- **MOVED:** OpcNodeAttributeCommand from Opc.UaFx.Client to Opc.UaFx.Services.

- **RENAMED:** “OpcNodeCommand” to “OpcNodeServiceCommand”.
- **RENAMED:** “OpcNodeAttributeCommand” to “OpcNodeAttributeServiceCommand”.
- **RENAMED:** OpcStatusCode members with “Bad”, “Good” and “Uncertain” prefixes like the OPC Foundation does use them to differ status codes into these three categories.
- **NEW:** Implemented additional ReadNode overloads to define a numeric Node identifier together with a namespace index and attribute to read.
- **NEW:** Implemented new ReadNodes methods to read multiple Nodes using a specific attribute.

1.6.5.2 (released 2017-02-07)

- **CHANGED:** Removed implicit interpretation of a numeric Node identifier encoded as string.

1.6.5.1 (released 2016-12-08)

- **NEW:** Implemented enhanced timeout handling for read history requests produced by the OpcClient. This is necessary to give the Server additional time to collect the historical data in case it will take more time than usual and the Server does provide successive continuation points until the data collection has been completed.

1.6.5.0 (released 2016-12-07)

- **NEW:** Enhanced Node History Navigation in case of a fixed time window (there StartTime and EndTime are not equals to MinDate). Therefore Node history values are collected on each page request as long as the Server provides a continuation point either until the Server does no longer provide a continuation point or until the page has reached its page size.

1.6.4.0 (released 2016-10-12)

- **NEW:** Updated IOpcNode interface definition in respect to the current public OpcNode API.
- **NEW:** Implemented new method Children() : IEnumerable<IOpcNode> in IOpcNode to determine physical children in the Node tree.

1.6.3.0 (released 2016-10-06)

- **NEW:** Implemented the new method OpcFileMethods.IsFileNode to determine if a Node is accessible in the manner of a FileType.
- **FIXED:** Out of memory issue when using OpcFile.ReadAllText and OpcFile.ReadAllLines.
- **NEW:** Improved performance of OpcFile.ReadAllText to no longer construct the text using the lines of the file, instead it now uses the binary stream of the file.
- **NEW:** Generally optimized the speed of reading and writing file contents by using the Client configured MaxByteStringLength.
- **FIXED:** Issue when browsing Nodes containing remote Nodes which led to the exception: 'Cannot cast an absolute ExpandedNodeId to a NodeId. Use ExpandedNodeId.ToNodeId instead.'

1.6.2.0 (released 2016-10-04)

- **NEW:** Implemented the new method CreateTempCertificate which is also used in cases where AutoCreateCertificate is set to true and where the certificate generator utiltity is not installed.

Therefore there is no longer a need to have the certificate generator utility implemented for testing / development besides OPC applications.
- **FIXED:** Issue with opening files for writing - the access mask has been verified in the wrong way.
- **NEW:** Implemented a new Session Resource Manager to dispose of any session related resources whenever a session times out or closes. Such a resource is the internal file handle when accessing file Nodes via OPC. In case the Client discards the connection to the Server upon any reason the Session Manager now releases all resources by the session acquired.

1.6.1.0 (released 2016-09-14)

- **NEW:** Implemented the OpcNodeManager.Browse method to support custom browsing in subclassing scenarios.

1.6.0.0 (released 2016-08-25)

- **NEW:** Excluded non-value attributes from automatically updating the source label of the value when writing Node values. Workarounds to use an explicit OpcValue are now obsolete.
- **NEW:** Introduced new derivate OpcBrowseNodeContext especially for Node browsing and reduced previous OpcNodeContext to a more generic implementation for further subclassing. The new OpcBrowseNodeContext does now also include the Node identifier of the Node from which the browsing operation has been originally introduced.
- **NEW:** Introduced OpcValue.As<T> method to retrieve the represented value as a specific type by converting the value to the type specified by T.
- **NEW:** Implemented the new OpcResult class to represent ServiceResult instances.
- **NEW:** Implemented the subclass OpcMethodContext of OpcContext to determine method call sensitive system information including the method Node being called and its target. This does also include storing the outcome of the callback routine invocation.
- **NEW:** Implemented file system accessibility support for OpcFileNode objects from the Client point of view. This does include a simple 1:1 API layer implemented by the OpcFileMethods class, a SafeOpcFileHandle to ensure allocated file handles are freed, the OpcFile class to implement an OPC based System.IO.File class like set of service methods, the OpcFileInfo class to implement an OPC based System.IO.FileInfo class like object and the OpcFileStream class to implement an OPC based System.IO.FileStream like object which is a derivated of the System.IO.Stream class and can therefore used by all StreamReader and StreamWriter classes provided by the .NET Framework.
- **NEW:** Implemented the new OpcPropertyNode and OpcPropertyNode<T>.
- **CHANGED:** The type OpcVariableValue does now require a generic type parameter for the type of value represented. The OpcVariableNode class has been adopted regarding this type parameter.
- **NEW:** Implemented the new OpcFileNode including its child Nodes like the method Nodes Open, Close, GetPosition, SetPosition, Read and Write.

1.5.11.7 (released 2016-07-28)

- **FIXED:** Issue with remote Node identifiers in OpcNodeId.
- **FIXED:** Issue with comparing Node identifiers of different types in OpcNodeId.

1.5.11.6 (released 2016-06-17)

- **NEW:** Implemented the Child method on OpcNodeInfo to retrieve the successors of a Node using its

OpcName. This does include the BrowseName, DisplayName and SymbolicName of a Node.
- **NEW:** Implemented the Parent method on OpcNodeInfo to retrieve the ancestors of a Node using its OpcName. This does include the BrowseName, DisplayName and SymbolicName of a Node.
- **NEW:** Implemented the Children method on OpcNodeInfo to retrieve successors of a Node of a specific OpcNodeCategory.
- **NEW:** Implemented the Parents method on OpcNodeInfo to retrieve ancestors of a Node of a specific OpcNodeCategory.

1.5.11.5 (released 2016-06-13)

- **NEW:** Implemented implicit cast operator for ExpandedNodeId's.
- **NEW:** Implemented additional Get method overload on OpcNamespace to retrieve Node namespaces using an Uri and Index.
- **FIXED:** Issue with ExpandedNodeId's in OpcValue's and simplified the handling of them.

1.5.11.4 (released 2016-05-30)

- **NEW:** Minimized lock conditions when reading / writing Nodes.

1.5.11.3 (released 2016-05-02)

- **NEW:** Demo license has been renewed to provide a new license period.

1.5.11.2 (released 2016-04-18)

- **FIXED:** Issue with monitored items when reading the initial value of a Node (the Server ran into a NullReferenceException in case of unfiltered monitored items).

1.5.11.1 (released 2016-03-31)

- **NEW:** Implemented the Changed event on OpcStatus.
- **FIXED:** Issue with the not synchronized OPC Status from SDK to the foundation stack (on OpcValue and OpcVariableNode).

1.5.11.0 (released 2016-03-23)

- **NEW:** OpcNamespace which represents the whole namespace information of a Node identifier (Namespace Index and Namespace Uri).
- **NEW:** IOpcNamespaceResolver which is implemented by OpcNodeManager, OpcClient and OpcServer. This interface does provide late bound namespace resolution after the OpcServer is started or the OpcClient is connected. Each OpcNodeId whose namespace has been resolved does provide the fully qualified URL of the namespace together with the Node identifier.
- **NEW:** OpcNodeId.Namespace property to provide the associated OpcNamespace.
- **CHANGED:** Renamed OpcAttribute.Historizing to OpcAttribute.IsHistorizing.
- **NEW:** OpcNode: Implemented Node change tracking through the properties HasPendingChanges and PendingChanges including the method IsChangePending.
- **NEW:** Implemented history support for OpcVariableNode and its subclasses.
- **NEW:** Enhanced constructor overloads of OpcStatusCollection to much simpler setup of a new

instance using OpcStatusCodes.
- **NEW:** HDA methods on the Client to read and update Node history.
- **NEW:** HDA interface on the Server to provide the required logic to handle historical data access.
- **NEW:** HDA services: IOpcReadNodesHistoryService and IOpcUpdateNodesHistoryService.
- **NEW:** HDA commands: OpcCreateNodeHistory, OpcDeleteNodeHistory, OpcDeleteNodeHistoryAtTime, OpcDeleteNodeHistoryModified, OpcReadNodeHistory, OpcReplaceNodeHistory and OpcUpdateNodeHistory. All of them are subclasses of the OpcNodeHistoryCommand.

1.5.10.0 (released 2016-02-03)

- **ADDED:** OPC Watch application to assembly archive.

1.5.9.1 (released 2016-01-25)

- **FIXED:** Issue with wrongly released session although it can be reused in case of a reconnect.

1.5.9.0 (released 2016-01-21)

- **FIXED:** Possible multithreading issues in session keep alive in OpcClient.
- **CHANGED:** The BreakDetected event is now being raised only if the reconnect timeout equals zero.
- **FIXED:** In case of releasing an OPC Client session which has already been disposed of in some cases it could lead to an undesired reconnect to the Server upon a thrown ObjectDisposedException by the already disposed Client session.

1.5.8.0 (released 2016-01-18)

- **NEW:** Updated the licensing logic.

1.5.7.1 (released 2016-01-14)

- **FIXED:** Issue with the minimum possible PublishingInterval in the OpcSubscription class.

1.5.7.0 (released 2016-01-09)

- **FIXED:** Issue with the null reference exception in OpcCertificateManager.CreateCertificate when passing an applicationUri as a null reference (Nothing in Visual Basic).
- **FIXED:** Issue with not uniquely generated OpcNodeId's for InputArguments and OutputArguments of OpcMethodNode instances. This led to the behavior that only the arguments of the most recently provided OpcMethodNode were published when browsing / reading the method argument properties.
- **CHANGED:** The OpcNodeId for InputArgument and OutputArgument Nodes of OpcMethodNode instances, for more information see the topic before.
- **CHANGED:** The OpcReadOnlyNodeCollection class does now provide IOpcNode instances instead of the internal foundation NodeState instances.
- **NEW:** An OpcNodeId can now be constructed from OpcName instances whose SymbolicName is a null reference (Nothing in Visual Basic). In this case the BrowseName is used instead of the SymbolicName.
- **NEW:** The OpcReadNodesRequest does now by default return both timestamps (source and Server).

- **NEW:** Implemented new OpcNodeManager methods: AddNode, RemoveNode and IsNodeAccessable.
- **FIXED:** Issue with duplicate ACEs some of the time in case of requesting an ACE which has been already added to an ACL.

1.5.6.8 (released 2015-12-17)

- **FIXED:** Issue with missed TimeStamp update on writing Node values.
- **FIXED:** Issue with missed TimeStamp data on reading Node values.

1.5.6.7 (released 2015-12-17)

- **NEW:** Implemented the ChangePassword method on OpcServerIdentity to change the user password after an identity has been created.
- **NEW:** Implemented the protected SystemContext property on OpcNodeManager to provide an alternative context within system internal changes can be performed.
- **NEW:** The OpcWriteNode class does now automatically set the source label timestamp when writing a Node.
- **CHANGED:** The OpcValueLabel is now no longer defined as a struct, instead it is now a class.

1.5.6.6 (released 2015-12-16)

- **NEW:** Implemented AddNode and RemoveNode in OpcNodeManager to support dynamically providing Nodes after CreateNodes has been called.

1.5.6.5 (released 2015-12-10)

- **NEW:** Added additional session information to OpcOperationContext calling IsNodeAccessable on OpcNodeManager.

1.5.6.4 (released 2015-12-09)

- **NEW:** Implemented IsNodeAccessable method on OpcNodeManager to customly specify whether a Node is accessible within the view.

1.5.6.3 (released 2015-12-08)

- **FIXED:** Issue with duplicate ACE's when manually adding additional entries on demand.

1.5.6.2 (released 2015-10-12)

- **FIXED:** Issue with reading and writing OpcValue instances and arrays as Node values on Client and Server sides.
- **FIXED:** Issue with anonymous user login on Server side when creating an OpcSession object.
- **NEW:** Implemented redirection in case the string value passed to the constructor (string, int) of the OpcNodeId class does indicate a numeric identifier. Now an OpcNodeId will be initialized with the numeric identifier instead of using the string value.

1.5.6.1 (released 2015-09-08)

- **FIXED:** Issue with Anonymous and UserName ACL's enabled at the same time.
- **NEW:** Implemented additional CreateCertificate overloads in OpcCertificateManager.

1.5.6.0 (released 2015-09-07)

- **CHANGED:** Renamed BuiltInType property to DataType on OpcAttributeInfo.
- **NEW:** Browsing the DataType attribute does now result into a member of the OpcDataType enumeration value.
- **NEW:** Implemented OpcValue as a surrogate of the DataValue class.
- **NEW:** Implemented the static CreateCertificate method on OpcCertificateManager to create certificates out of the box.
- **NEW:** OpcStatus class does not provide the new OpcStatusCode enumeration as Code.
- **NEW:** Implemented support for OpcDataTypes: ExtensionObject, Value and Variant.
- **CHANGED:** Renamed DataValue to Value in OpcDataTypes.
- **NEW:** Implemented support to directly start browsing using a method Node.
- **CHANGED:** Moved OpcDiscoveryClient to Opc.UaFx.Client namespace.
- **FIXED:** Issue with NullReferenceException when noncached Nodes are browsed.
- **NEW:** Implemented support using a null reference for an application URI to create certificates.
- **NEW:** Implemented a new abstraction layer for different OPC Server types (OpcServerBase).
- **NEW:** Implemented a new OPC Server specialization: OPC Discovery Server.
- **FIXED:** Issue in OPC Client with accessing the internal session instance although it has been disposed of.

1.5.5.3 (released 2015-08-25)

- **NEW:** Simplified the use of OpcVariableClass in Read / Write callbacks through determining the status and timestamp (if no other has been specified) from the Node instance.

1.5.5.2 (released 2015-08-24)

- **FIXED:** Issue with Anonymous and UserName ACL's enabled at the same time.
- **NEW:** Removed OpcAccessControlMode member 'Default' and replaced its use with Blacklist.

1.5.5.1 (released 2015-08-24)

- **FIXED:** Issue with Null DataType in VariableNodes.

1.5.5.0 (released 2015-08-24)

- **NEW:** Implemented different callback properties to read / write OpcNode attributes.
- **NEW:** Implemented protected virtual methods to read / write OpcNode attributes.
- **NEW:** Implemented specific callback properties to read / write OpcVariableNode attributes.
- **NEW:** Implemented protected virutal methods to read / write OpcVariableNode value attributes.
- **NEW:** The OpcSession class does now provide the properties UsedIdentity and SuppliedIdentity.
- **NEW:** Implemented the Update method on OpcStatus.
- **NEW:** The OpcContext does now provide an Identity property.

- **NEW:** Implemented the new OpcMonitoredItem class in the Server namespace.
- **NEW:** OnMonitoredItemCreated/Deleted/Modified does now provide non-foundation classes.
- **CHANGED:** Moved Identity and ImpersonationContext classes from Opc.UaFx.Server to Opc.UaFx namespace.

1.5.2.0 (released 2015-06-18)

- **NEW:** Excluded the auto update of the endpoint before the connect in cases where a preferred endpoint is defined in OpcClient.
- **NEW:** Implemented new endpoint identity security on OpcSecurity to enable / disable different endpoints for one or more ACEs. Additionally renamed some existing methods.
- **NEW:** Implemented StateChanged events in OpcClient.

1.5.1.1 (released 2015-06-16)

- **NEW:** Implemented InvokeService method in OpcClient to invoke service routines using a try-catch-finally block to ensure that the Client's state property does also handle its connection state in case of an exception, which indicates that the connection has been timeout, lost, etc.
- **FIXED:** Issue with null references as DisplayName and BrowseName in OpcReferenceDescription.

1.5.1.0 (released 2015-06-15)

- **NEW:** Implemented the SaveCertificate method on OpcCertificateManager.
- **NEW:** Reworked the complete argument binding to also support out and ref parameters in OpcMethodNode.
- **NEW:** Implemented ByRef Type support in OpcMethodNode.
- **NEW:** Implemented the local Server CertificateValidationFailed event to support custom failed certificate validation actions.
- **NEW:** Implemented the new OpcNodeInfo derivated from OpcMethodNodeInfo to browse the method Nodes arguments.
- **NEW:** Implemented the new OpcCertificateValidationFailedEventArgs and OpcCertificateValidationFailedEventHandler.
- **NEW:** Implemented the DataType property on OpcDataVariableNode.
- **NEW:** Implemented InputArguments and OutputArguments properties on OpcMethodNode class. Additionally extended the support of OpcArgument instances with the OpcArgumentAttribute on custom delegates.
- **NEW:** Implemented support for null references in all implicit cast operators which are working with reference types.

1.5.0.0 (released 2015-06-10)

- **NEW:** Implemented logic to generate Server fulfilled configuration on start.
- **NEW:** Completely reworked the endpoint selection using the OpcDiscoveryClient and custom preferred OpcSecurityPolicy.
- **NEW:** Improved the auto complete of the configuration depending on its application type.
- **NEW:** Introduced transport profile information.
- **NEW:** Moved the configuration of UserTokenPolicies to OpcServer on startup and improved the transparency of the IsEnabled property.

- **FIXED:** Issue with empty certificate store paths.
- **NEW:** Completely reworked the Node referencing mechanism to provide more information while building the Node references and to also remove OPC internals from the user API.
- **NEW:** Completely reworked existing Node implementations and moved their base class reference to an internal reference to improve the experience with the framework and become more CLSCompliant by also providing more useful APIs.
- **NEW:** Implemented support for discovery Server base address configuration.
- **NEW:** Improved configuration defaults, also when loading configuration through one of the Load methods. Additionally implemented default support for DiscoveryServerConfigurations.
- **NEW:** Simplified the API further.
- **NEW:** Introduced replacement for StatusCode → OpcStatus.
- **NEW:** Reworked OpcDiscoveryClient from factory methods to constructor use to provide a fluent interface between all Server and Client classes.

1.4.0.0 (released 2015-06-09)

- **NEW:** Merged Opc.ClientFx.Advanced and Opc.ServerFx.Advanced together to Opc.UaFx.Advanced.

1.0.0.0 (released 2015-01-11)

- **NEW:** Created Opc.ClientFx.Advanced and Opc.ServerFx.Advanced.

Table of Contents