OPC Xi Communication and its Configuration

July 2010   Advosol Inc.

Introduction

The OPC Xi specification defines a .NET interface for the data exchange between local and remote automation systems and uses Microsoft WCF (Windows Communication Foundation) for the communication.
Xi combines the functionality of the OPC DA, A&E and HDA into a single interface and is designed to make the implementation of wrappers for classic OPC servers simple and efficient.
Basing the Xi specification on WCF keeps the specification and the server/client implementation simple. WCF handles all communication details and options.
Xi is designed for communication with high performance and high security. This is achieved by using WCF in specific ways and grouping the Xi functionality so that sensitive data can be communicated with high security while for other data is transferred at lower, more efficient security levels.

This whitepaper intends to give an overview of the design concept with the available features and restrictions related to certain configuration options.

WCF Basics

Microsoft designed WCF as a unified programming model for distributed applications combining the features of interfaces for inter-process, LAN and Internet communication. WCF is part of the .NET Framework and provides the basis for communication between components, applications and systems.
WCF defines a .NET interface with contracts:

ServiceContract	The ServiceContract is assigned to a .NET interface (class description).
OperationContract	An OperationContract is assigned to each method that is to be exposed.
DataContract	Data classes used in the OperationContract methods.

The service developer implements a WCF service with one or multiple ServiceContracts and WCF handles the communication according the contract definitions.

The WCF SvcUtil utility creates client proxy code from the service contract definitions. The proxy code provides classes with basically the methods as they are defined in the server interface definition. The client simply invokes these methods to communicate with the server. The communication details on the server and client side are handled by WCF.

WCF Configuration

The WCF communication is configured in endpoint definitions. A WCF service can have any number of endpoints. The configuration for each endpoint defines:

- ServiceContract. The service class accessed through this endpoint.
- Binding. The communication protocol.
- Address. A unique address within the service.

The WCF configuration can either be defined in code or in the applications configuration file.
The server and the client must define endpoints with the exact same configuration.

The Binding has the most significant effect on how WCF communicates. It controls the aspects:

- The suite of WS-* protocols, e.g. WS-Security, WS-ReliableMessaging etc.
- The message encoding such ad XML, MTOM, binary
- The transport protocol, such as HTTP, TCP, NamedPipe

WCF Configuration in Xi

Usually the Xi server configuration is defined in the application configuration file, but the Xi server could also define the WCF configuration in its code.

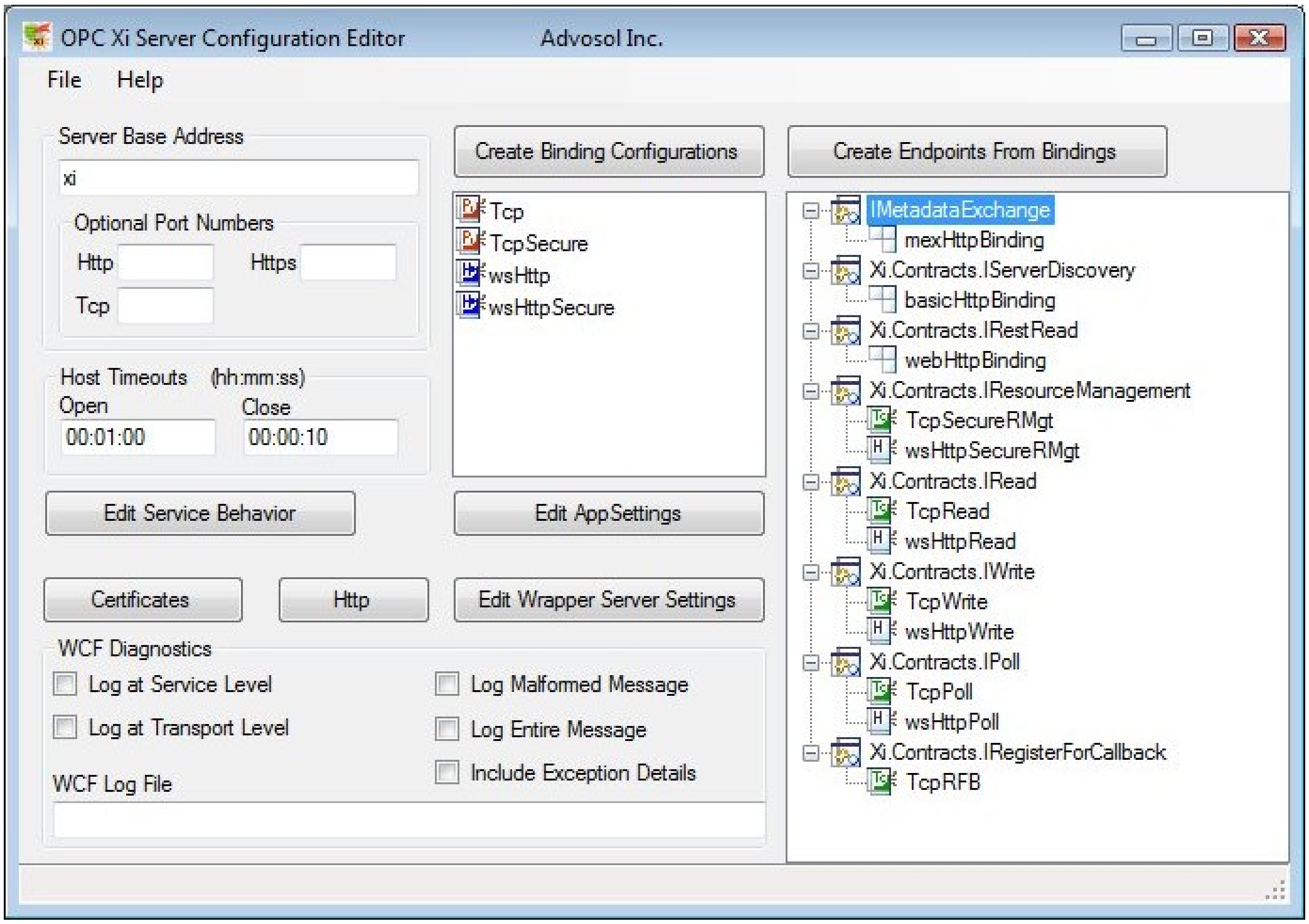
The Xi server provides features that allow the clients to retrieve the WCF endpoint configuration from the server. The communication configuration for Xi clients is reduced to the selection of the preferred endpoint among the endpoints configured in the server.

To achieve this simplification all Xi servers must have the endpoint for the IServerDiscovery service contract configured for:

- Address: *ServerDiscovery*
- Binding: *basicHttpBinding* with default settings.

For all other ServiceContracts the Xi server can configure any number of endpoints with any of the WCF bindings.

The following screenshot of the Advosol Xi Configuration Editor shows a typical Xi server endpoint configuration.


OPC Xi Server Configuration Editor
Advosol Inc.
File
Help
Create Binding Configurations
Create Endpoints From Bindings
Edit Service Behavior
Edit AppSettings
Certificates
Http
Edit Wrapper Server Settings

Server Base Address: xi

Optional Port Numbers: Http, Https, Tcp

Host Timeouts (hh:mm:ss) Open: 00:01:00 Close: 00:00:10

Tcp
TcpSecure
wsHttp
wsHttpSecure

IMetadataExchange
- mexHttpBinding

Xi.Contracts.IServerDiscovery
- basicHttpBinding

Xi.Contracts.IRestRead
- webHttpBinding

Xi.Contracts.IResourceManagement
- TcpSecureRMgt
- wsHttpSecureRMgt

Xi.Contracts.IRead
- TcpRead
- wsHttpRead

Xi.Contracts.IWrite
- TcpWrite
- wsHttpWrite

Xi.Contracts.IPoll
- TcpPoll
- wsHttpPoll

Xi.Contracts.IRegisterForCallback
- TcpRFB

WCF Diagnostics
- [ ] Log at Service Level
- [ ] Log at Transport Level
- [ ] Log Malformed Message
- [ ] Log Entire Message
- [ ] Include Exception Details

WCF Log File

The Xi client retrieves the endpoint configuration from the server and selects among the available endpoints. This can be done in different ways, e.g.:

a) The client application shows the endpoint details in a dialog and the operator chooses the endpoint
b) The client is built for a specific server and knows the names of the endpoints to be used.
c) The client has a preference for a specific binding (e.g. TCP) and security level and has code to select the best matching available endpoint.

OPC Xi Service Structure and Security

The Xi specification defines a WCF service with the ServiceContracts:

IServerDiscovery	Server discovery methods
IResourceManagement	Server management methods.
IRead	Read data
IWrite	Write data
IPoll	Poll data changes
IRegisterForCallback	Request data change callbacks

The Xi features are designed and grouped into contracts to achieve high security with high performance.

The server management methods in the **IResourceManagement contract** are most likely to communicate sensitive data and require high security. Performance is less of an issue because these methods are typically used mainly in the startup phase.
It is recommended to use only secure bindings with the IResourceManagement endpoints. Many Xi servers are implemented to actually accept only secure bindings.

The methods in the **IRead, IWrite and IPoll contracts** transfer data values. Performance is important because these methods are likely used periodically. Xi uses only handles with the data values, no interpretable object identifiers. This keeps the data size small and makes the data difficult to interpret even if an eavesdropper manages to capture the communicated data. Applications that don't have top security requirements can transfer data without encryption and gain performance.

Xi servers can restrict write access without writing special code by either:

- Configuring no IWrite endpoint to disable write access completely
- Configuring only IWrite endpoints with secure bindings to restrict write access to authenticated clients
- Configuring only IWrite endpoints with namedPipe binding to restrict write access to local clients

WCF Communication

The WCF communication is essentially determined by the Binding configuration. Each Binding uses one of the WCF supported Transports:

HTTP Transport	HTTP is a request/response protocol between clients and servers. The client sends a request to a server, which listens for client request messages. When the server receives a request, it returns a response, which contains the status of the request and optional data. In WCF, the HTTP transport binding is optimized for interoperability with legacy non-WCF systems. If all communicating parties are using WCF, the TCP or Named Pipe transports are faster.
TCP Transport	TCP is a connection-based, stream-oriented delivery service with end-to-end error detection and correction. Connection-based means that a communication session between hosts is established before exchanging data. A host is any device on a TCP/IP network identified by a logical IP address. TCP provides reliable data delivery and ease of use. Specifically, TCP notifies the sender of packet delivery, guarantees that packets are delivered in the same order in which they are sent, retransmits lost packets, and ensures that data packets are not duplicated. Note that this

	reliable delivery applies between two TCP/IP nodes, and is not the same thing as WS-ReliableMessaging, which applies between endpoints, no matter how many intermediate nodes they may include. The WCF TCP transport is optimized for the scenario where both ends of the communication are using WCF. This binding is the fastest WCF binding for scenarios that involve communicating between different machines. TCP provides duplex communication and so can be used to implement duplex contracts, even if the client is behind network address translation (NAT).
Named Pipe Transport	A named pipe is an object in the Windows operating system kernel, such as a section of shared memory that processes can use for communication. A named pipe has a name, and can be used for one-way or duplex communication between processes on a single machine.

Some of the available WCF Bindings are:

NetNamedPipeBinding	The NetNamedPipeBinding is an appropriate choice for on-machine communication. It uses named pipes for message delivery and by default transport security and binary message encoding. The default configuration for the NetNamedPipeBinding is similar to the configuration provided by the NetTcpBinding, but it is simpler because the implementation is only meant for on-machine use and consequently there are fewer exposed features.
NetTcpBinding	The NetTcpBinding is an appropriate choice for communicating over an Intranet or the Internet. It uses TCP for message delivery and transport security and binary message encoding. The default configuration for the NetTcpBinding is faster than the configuration provided by the WSHttpBinding. Reliable messaging is off by default. More generally, the HTTP system-provided bindings such as WSHttpBinding and BasicHttpBinding are configured to turn things on by default, whereas the NetTcpBinding binding turns things off by default so that you have to opt-in to get support, for example, for one of the WS-* specifications. This means that the default configuration for TCP is faster at exchanging messages between endpoints than that configured for the HTTP bindings by default. TCP is a bi-directional protocol. Client and server can send requests on the same connection. This considerably simplifies bidirectional communication through routers with network address translation (NAT). The NetTcpBinding requires a unique port number for each connection. For communication through firewalls the port number has to be added in the firewall configuration.
BasicHttpBinding	The BasicHttpBinding uses HTTP as the transport for sending SOAP 1.1 messages. A service can use this binding to expose endpoints that conform to WS-I BP 1.1, such as those that ASMX clients access. Similarly, a client can use the BasicHttpBinding to communicate with services exposing endpoints that conform to WS-I BP 1.1, such as ASMX Web services or services configured with the BasicHttpBinding. It uses a "Text" message encoding and UTF-8 text encoding by default.

WsHttpBinding	The WSHttpBinding is similar to the BasicHttpBinding but provides more Web service features. It uses the HTTP transport and provides message security, as does BasicHttpBinding, but it also provides transactions, reliable messaging, and WS-Addressing, either enabled by default or available through a single control setting.
WsDualHttpBinding	The WSDualHttpBinding provides the same support for Web Service protocols as the WSHttpBinding, but for use with duplex contracts. WSDualHttpBinding only supports SOAP security and requires reliable messaging. This binding requires that the client has a public URI that provides a callback endpoint for the service. This is NOT the case for clients behind routers with network address translation (NAT). For duplex connections to work through a NAT the NAT needs to be configured with forwarding rules. Such configuration is NAT specific and many end users either cannot or do not want to change their NAT configuration for a particular application. Therefore this binding is not usable for duplex communication in most cases. A dual binding exposes the IP address of the client to the service. The client should use security to ensure that it only connects to services it trusts.

Collections

.NET supports multiple kinds of collections such as arrays and lists. Generic lists offer convenient features not available in arrays but different developers have different preferences.
WCF communicates only one kind of collections. The SvcUtil utility has an option that determines if it creates proxy code with either arrays or generic lists. However, all collections are created in the same type, even if the server uses a mix of arrays and lists.
There is another consequence that may not be obvious. Let’s look at the ReadData method. It returns a collection of current data values. These values can be simple type values of collection type values. The selected kind of WCF collection handling also determines how the type of the collection the client receives the collection type values of Xi objects.
Example: The client executes a read for 3 Xi objects (OPC DA items). These objects have array type values. If the WCF proxies are created for generic list collections then the client gets a read result with a list of 3 values with each value being a list type collection.

Buffer Sizes

The WCF default buffer size definition in the bindings is 64kB. This is often too small and needs to be enlarged. Some bindings have multiple size settings and in most cases these must be set to the same value.
Another size limitation that may be overlooked is the maximum number of WCF objects in any message (maxItemsInObjectGraph). The objects are not always easy to count. As an example, the AddDataObjectToList method result has 10 serialization objects for each Xi object. In messages with array values, every array entry counts as a separate object. This setting can be defined in the endpoint behavior or the server behavior. WCF uses the larger of the two for each endpoint. To eliminate issues with this property it can be set to Int32.MaxValue (2147483647). The overhead this causes regarding performance and memory use is hard to determine.

Xi Communication Performance Optimizations

Extensive Xi performance tests revealed that the WCF data serialization is much slower for object type collections than it is for fix type collection. Explicitly, the serialization of an object[] with Int32 values is slower than the serialization of an Int32[].
OPC servers with large number of items often use mainly Int32 or Double values. The performance could be significantly increased by communicating the data in Read, Poll and dataChange callbacks in a special class. This class basically has the data separated into arrays:

```
double[] DoubleValues;              Double and Single values
uint[] DoubleStatusCodes;
DateTime[] DoubleTimeStamps;
uint[] UintValues;                  32/16/8-bit integer values
uint[] UintStatusCodes;
DateTime[] UintTimeStamps;
object[] ObjectValues;              All other type values
uint[] ObjectStatusCodes;
DateTime[] ObjectTimeStamps;
```

The Xi applications are usually not confronted with this optimization. The base layer Xi server code builds an instance of this class from a .NET object collection and communicates it. The Xi client base layer code transforms the data back into the more convenient form:

```
List<ReadValue> data;
```

With the ReadValue class basically having the members:

```
object Value;
uint StatusCode;
DateTime TimeStamp;
```

Polling or Callbacks

In OPC DA the use of data change callbacks is essential for high communication performance. The only alternative to data change callbacks is for the client to periodically read all item values. To limit the latency the read has to be executed frequently and likely only a few of the read values actually have changed.

In OPC Xi callbacks are only possible with bindings that have duplex capability. Practically callbacks are only possible with NetTcpBinding or NetNamedPipeBinding. WsDualHttpBinding requires the client to have a public URI and this is not often the case. However, NetTcpBinding can be used in all network configurations as long as both sides are .NET based.

OPC Xi supports a Poll mechanism for cases where callbacks are not possible. With a poll request the client can instruct the server to reply with the changed data values. The overhead compared to data change callbacks is minor, only the communication of the short poll request message.

Some commercial Xi base layer components handle polling/callbacks internally so that the application always gets callbacks, independent of the actual communication mode.