

WCF Concepts

by Sam Phoenix

FILE	2644_SAM_PHOENIX_WCF_CONCEPTS_94244_1934001109.DOCX (41.38K)		
TIME SUBMITTED	30-JAN-2018 11:51PM (UTC+0000)	WORD COUNT	2269
SUBMISSION ID	82596924	CHARACTER COUNT	14506

WCF Concepts

DISTRIBUTED ENTERPRISE APPLICATIONS

Sam Phoenix

P151028 | LECTURER: TIM BASHFORD

WCF CONCEPTS

I. Introduction	2
II. Predominant rationale for introducing WCF	3
A. COM (Component Object Model)	3
B. DCOM (Distributed Component Object Model)	3
C. .NET remoting	3
D. Web Services	3
E. WCF	4
III. How to host a WCF service	5
A. The ABCs of WCF	5
B. Principles and process of metadata exchange	6
IV. The four major contract types.....	8
A. Service	8
B. Operation	8
C. Data	8
D. Fault.....	8
V. Supported exchange patterns.....	9
A. Request-response	9
B. One-way	9
C. Duplex	9
D. Asynchronous	9
E. Peer to peer (P2P)	9
VI. How WCF enables and supports interoperability	11
VII. Conclusion.....	12
VIII. References	13

I. Introduction

This report will discuss the Windows Communication Foundation (WCF), including how it was developed, its features and how it can support different clients.

II. Predominant rationale for introducing WCF

To understand why WCF was developed, it is necessary to look back at Microsoft's development history[1]:

A. COM (Component Object Model)

In the early days of Windows, COM was developed as a means of communicating between devices. Serial ports were labelled COM1, COM2 etc.

COM is a universal model that can allow different components written in different languages to communicate with each other. While it works well with local machines, it cannot effectively deal with remote components.

B. DCOM (Distributed Component Object Model)

As the internet became more widespread, and demand for remote handling increased, DCOM was developed. This combines the COM model with the network protocol, and includes features like security, location independence and reliability. The disadvantages of this are that it only works across Windows systems and is tightly coupled with the server providing the service – if there is even a small change to the server application, every device communicating with it must update their software.

C. .NET remoting

.NET was developed as a step towards software as a service (SaaS) and is a way to provide distributed applications. It provides a modular model using a combination of components that can be replaced at will and extends the communication capacity of DCOM to reach devices on different networks.

Like DCOM, .NET remoting suffers the drawback of only being usable on Windows based machines. It also struggles with load-balancing; it can not tell if a server is too busy to process an application request.

D. Web Services

Web services are the first step towards service-oriented development, which is a set of guidelines for developing systems that are loosely-coupled; the main functionality of the program remains the same, so any applications wishing to communicate with it do not need to update every time the host application changes.

This is also a step away from proprietary protocols like .NET and towards making use of universally accepted protocols such as HTTP, XML, and SOAP. This means that the host application can be accessed by non-Windows devices.

E. WCF

Development on WCF started at about the time .NET was released, and includes compatibility with Microsoft's previous models, building upon and extending their functionality. It has been designed to be service-oriented and future-proof, including many ways to communicate with different systems and to be hosted.

WCF is a replacement for earlier Microsoft web technologies[2] discussed above. It is designed to be more flexible in its approach to web services, and uses a more efficient means of serialising data (DataContractSerializer as opposed to XmlSerializer).

III. How to host a WCF service

In order to use a WCF service, it needs to be hosted. There are four ways of doing this[3]:

- **IIS hosting:** IIS stands for Internet Information Services, which has been in use for many years. Its best feature is automatic processing of service activation, and offers other services such as health monitoring, idle shutdown and process recycling. This is the same method used by earlier ASP.NET web services, but can only use HTTP.
- **Self-hosting:** This requires additional code to be developed so it can be used, but is the most flexible as it can be hosted in a variety of applications. It also offers more choice of protocol than IIS. It is most often used for console and Windows Forms or Windows Presentation Foundation (WPF) applications, which are installed on a local machine but need to connect to an external server.
- **WAS hosting:** WAS stands for Windows Activation Service. It offers the same features as IIS, but also offers extra configuration and protocol support. It works by extending on IIS, making it more flexible and therefore accepting of other protocols such as TCP.
- **Windows Service hosting:** If the application is hosted on a local computer and is designed to run in the background, it is best to take advantage of existing Windows services, which are supported by all modern versions of Windows. It can be configured to run when Windows starts and perform background tasks (like Scheduler) without any other user intervention.

A. The ABCs of WCF

When planning to use a WCF service, there are three main questions that need to be asked[4]:

1. Where is the service located from the client's perspective?
2. How can a client access the service?
3. What is the functionality being provided to the client?

Answering these questions come in the form of “the ABC of WCF”, which are described below:

- **Address:** A WCF provides a URI that can be used to get to the service, and an Identity property which handles the security identity, as well as more advanced addressing information. These can be found in the EndpointAddress class.
- **Binding:** This handles the protocols and message formats used by the client to communicate with the service, as well as security requirements for those protocols (i.e. SSL for https or message security for SOAP). It is handled by the abstract class Binding.
- **Contract:** This defines what functionality the service provides, including interfaces and data being passed between service and client. It specifies what operations can be called, what form the message will take, the input parameter or data types and the kind of message processing that will take place.

The ABCs above form the basis of endpoints, which provide access to the functionality of a service[5].

B. Principles and process of metadata exchange

Metadata is used by WCF services to provide a means for clients to communicate with the host application. It can be generated by the host to describe the services, addresses, contracts etc. that the client will need to be able to connect with the application and access its services. It uses the ServiceEndpoint class and the System.ServiceModel namespace – the latter needs to be included as a reference in any application using WCF for this reason.

Metadata **export** is the process of “describing service endpoints and projecting them into a parallel, standardized representation that clients can use to understand how to use the service” [6]. It generates policy data that describes the functionality and requirements of endpoints, including the associated contract and binding information.

Publishing metadata makes it available to the client using protocols such as HTTP/GET requests. It involves publishing metadata endpoints, which are similar to service endpoints in that they contain the ABCs discussed above.

Retrieving metadata is done by the client via protocols such as HTTP. An address is provided for the client to retrieve this metadata for later use.

Once the data is retrieved, it must be **imported**. This process involves generating the service information for a client to connect to the host, including the address, binding and

other details needed to access the service. This data can be imported to the `app.config` file so the client can easily connect to the host.

IV. The four major contract types

Contracts hold information about the functionality of the service being provided. There are four main types; these are discussed below.

A. Service

A service contract provides information about the operations in a service, including data types, location and protocols being used. They are constructed using a combination of interfaces, classes and attributes to define the structure of, and implement, a service. This information can be exported as metadata for use by a client.

B. Operation

Operation contracts specify the individual operations to be performed by the service. They are defined with the `OperationContract` attribute, which is used to mark a method within a class as a service operation. The `ServiceContract` attribute is then used to group these operations together to form part of the service contract.

C. Data

Data contracts are an agreement between the host and client on the data to be exchanged. It uses `DataContract` and `DataMember` attributes to specify which data will be shared with the client, and the resulting information is converted to XML so a client can use it – XML is used as it is a standard format that any system can read. It uses the `DataContractSerializer` to perform this conversion, which is similar to the `XmlSerializer` used by ASP.NET web services.

D. Fault

The fault type is responsible for error handling in WCF applications. In managed applications, these are represented by Exceptions; in SOAP applications, they are processed as SOAP faults. WCF is capable of handling both types, with custom error messages declared in a fault contract that is included in metadata.

V. Supported exchange patterns

A. Request-response

This is the default exchange used by WCF. It involves the client making a request call to the service and waiting for a response from the server. It can be performed synchronously – where the client will not proceed until it receives a reply – or asynchronously – where the client will issue a request and continue working, using another thread to listen for a reply.

B. One-way

A one-way request happens when the client sends a call to the host, but does not expect a reply. This is used when the client is not affected by the result of the operation being performed in response to this request. It can be used in conjunction with duplex communication, where the host sends any error messages to the client via a callback function.

C. Duplex

Communication usually happens one message at a time. This is especially noticeable in request-response, where one message is sent – the request – then a second message, the reply, is sent later. This form of communication is known as half-duplex. Full duplex occurs when messages are sent by both endpoints simultaneously. It is set up using two interfaces – one on the host to send messages, and another on the client to send calls to the host, which includes a callback method for receiving messages from the host.

D. Asynchronous

Usually, when a message is sent by a client, it needs to wait for the response before performing the next task. Asynchronous communication makes use of threading, where the main thread is used to run through tasks, with another thread used for background services such as listening for a response from the server. This enables the client to continue through a task list while waiting for a response, which reduces processing time.

E. Peer to peer (P2P)

P2P networking refers to the way data is passed between computers on a network. Most networks are managed by a server which hosts applications that can be accessed by clients. P2P, on the other hand, has no centralised server that provides services; instead, computers on a network host and access each other's applications, providing and receiving services without the need of, or complementing, a central server.

Common uses of P2P include file sharing, chat and game applications, and real time news feeds, as well as Bitcoin mining and other computationally expensive tasks.

A collection of P2P computers on a network is called a mesh, and each can be identified with a unique ID. Each endpoint on this mesh is known as a node, and there can be multiple nodes in an application that are part of different meshes.

VI. How WCF enables and supports interoperability

Interoperability is the concept of cross-platform support; it recognises that not every client that requests services from a server is a Windows machine.

In earlier development models, proprietary protocols such as COM and .NET were used to communicate between devices. This way of working assumed that every system had use of these, but some clients run MacOS or Linux, which use their own means of communicating between others running those systems.

Web services made progress by providing services using standard protocols, such as HTTP and SOAP, converting the data to XML format which is readable by all systems. WCF expands on this concept, in one way by offering support for non-HTTP protocols such as TCP. It also provides support for other, newer endpoints that do not use SOAP or XML data; REST is a newer service model that is a more efficient means of providing services, and JSON is a more compact data format with cleaner code, as opposed to XML, which can be sent to AJAX-enabled clients.

VII. Conclusion

WCF is a means of providing service-oriented communication between clients and hosts, encompassing all previous development tools provided by Microsoft and expanding on these. The result is an extremely flexible model that can supply services to all manner of clients using a variety of means.

VIII. References

- [1] M. V, 'Programming with WCF: History of WCF'. [Online]. Available: <https://programmingwithwcf.blogspot.co.uk/2010/06/history-of-wcf.html>. [Accessed: 30-Jan-2018].
- [2] S. Orientation, 'What's the Difference between WCF and Web Services? - CodeProject', 2000.
- [3] 'WCF hosting'. .
- [4] 'A Beginner's Tutorial for Understanding Windows Communication Foundation (WCF) - CodeProject'. [Online]. Available: <https://www.codeproject.com/articles/406096/a-beginners-tutorial-for-understanding-windows>. [Accessed: 29-Jan-2018].
- [5] 'Endpoints: Addresses, Bindings, and Contracts | Microsoft Docs'. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/endpoints-addresses-bindings-and-contracts>. [Accessed: 30-Jan-2018].
- [6] 'Metadata Architecture Overview | Microsoft Docs'. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/metadata-architecture-overview>. [Accessed: 30-Jan-2018].

WCF Concepts

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

55/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1

.net isn't really a protocol - though it contains a few, like MSMQ.

PAGE 6



Comment 2

By convention, this is written:

Internet Information Services (IIS)

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

GRADING FORM: SMU GENERIC_GJ

SAM PHOENIX**55****RESEARCH-INFORMED LITERATURE**

The latter portion of your report is in desperate need of some inline citations - I'm almost certain you'll have read around the topic for the purpose of constructing this information, but much of it's entirely uncited. In general, the sources you've used aren't of high quality - Microsoft are reasonably authoritative on WCF (albeit accepting their inevitable bias), but blogs and code submission sites are not academically credible sources. The third source is incomplete in the reference list to the point of being unusable. Remember that the purpose of your reference list is to allow someone to find and read the sources within. I have a strong suspicion that many of these issues are the result of not having spent as much time as you'd have liked in constructing this work, not a lack of understanding. I'm more than happy to talk you through this, if you'd like, however, to help you avoid issues with this in your upcoming submissions.

2

KNOWLEDGE AND UNDERSTANDING

-  Fairly good evidence of knowledge and understanding - a few very minor errors in places. The report itself is a little brief, however - it'd be nice to see a little more detail in the key areas.

14

ANALYSIS

-  A lack of evidence of synthesis of material in general, as a result of a relatively low number of sources and citations. Try, when constructing an argument, to read a number of sources on a particular topic, write your argument (ideally after stepping away for a few minutes to avoid repeating the last thing in your head), then cite everything you read, e.g. [1][2][3][4]. You have some concluding remarks in a specific section which is nice to see, but you could definitely take this a little further - consider including short subsection conclusions where appropriate - these will actually help you flow your document together.

14

PRACTICAL APPLICATION AND DEP.

-  Some fairly good consideration of practical application. Your section on hosting in particular is good. To improve this remark, I'd suggest considering examples, either in code, or through text. In your ABC section, what does a URI look like? What kinds of bindings are supported? Etc.

18

SKILLS FOR PROFESSIONAL PRAC.



You write very well and have a pleasant written style, with good spelling, punctuation and grammar. The report is nicely presented. You might consider including a few figures to help illustrate your points in the relevant/appropriate places.

7