**WCF 개요**

 WCF는 분산 컴퓨터 작업, 광범위한 상호 운용성 및 서비스 지향성과 관련한 직접적인 지원을 위하여 개발된 프레임워크이다.



**분산컴퓨팅**

 WCF는 마이크로소프트의 그동안의 분산 컴퓨팅 기술을 하나로 통합 해 지원하고 단일화 된 프로그래밍 모델을 지원한다. WCF는 ASMX, WSE, MSMQ, Remoting, COM+, DCOM등의 기술과 같이 사용하는 것이 가능하도록 설계되었다. WCF를 통한다면 이러한 복잡한 기술들을 하나로 묶어 사용하는 것이 한결 쉬워 진다. 프로토콜, 데이터의 변환(Transformation) 과 전송(Transmission)은 WCF에 의해서 제공 된다.



**상호운용성**

 위의 개념을 전개해 실제 적용을 하면 상호 운용성 문제를 언급 해야만 하는데 아래와 같이 개념화 할 수 있다.







다른 Process, 다른 Window, 다른 웹서비스 사이에 WCF를 통해 상호 운용이 가능 하다. 같은 Window의 네트워크 안에서는 마이크로소프트의 기술로 통합이 가능하겠지만 이기종 운영체제 사이에서는 Vendor들이 동의하는 표준에 따라야 한다. 다음은 Webservice 와 관련 Protocol Stack 인데 WCF를 이를 그대로 수용한다.



WCF에 사용되는 표준은 기능별로 분류를 하면 다음과 같다. 관련 표준은 주로 Vendor들에 의해 제안되고 OASIS 와 W3C에 의해 관리가 된다.



XML 과 XSD Schema는 SOAP, WSDL, WS\*- 스펙을 정의하는데 핵심 역할을 한다. XML Digital Signature 와 XML Encryption는 XML Message를 안전하게 보호하기 위한 표준이 된다.



가장 중요한 SOAP 표준은 메시지 프로토콜의 표준이고 많은 Vendor들에 의해 채용되어 사용되고 있다. WS-Addressing은 Message와 Service의 EndPoint를 지정하기 위해 사용되는 Protocol이다. WS-Addressing 표준은 SOAP 메시지를 Transport Layer로부터 Decoupling 시킨다. 이로 인해 비동기 전송이나 메시지 처리가 가능하다. MTOM (Message Transmission Optimization Mechanism)은 SOAP message 내의 이진 데이터를 어떻게 효과적으로 전송하느냐에 관한 표준이다. WS-Transfer 표준은 SOAP을 통해서 리소스에 접근해 조작하기 위한 것으로 리소스에 대한 CRUD 작업을 가능하게 한다. WS-ResourceTransfer 표준은 WS-Resource 표준의 확장이다. WS-Enumeration 표준은 Large ResultSet 내에서 원하는 값을 탐색하기 위한 표준이다. WS-EventNotification은 분산 이벤트와 notification을 지원하기 위한 표준이다. 이를 통해 전송 프로토콜에 독립적인 publish/subscribe scenario를 가능하게 한다.



WSDL은 웹서비스를 기술하기 위한 XML 기반의 표준이다. WS-Policy 표준은 웹서비스의 Policy를 기술하기 위한 일반적인 문법 구문과 Model을 제시한다. WS-MetadataExchange 표준은 Webservice Endpoint를 검색하기 위해 지원되는 표준으로 Webservice 문서의 Discovery를 쉽게 한다.



WS-Security는 SOAP 메시지의 Encrytion과 Signing을 담당 하는 Open Format을 제공하고 이를 위해 SAML, X.509, Username Token Profile, Keberos Token을 이용한다. 보안 통신을 제공하기 위해 WS-Security를 토대로 하는 확장을 정의하고 특히, 보안 컨텍스트를 설정하고 공유하며 보안 컨텍스트에서 세션 키를 파생하기 위한 메커니즘을 정의한다. WS\_Trust 스펙은 보안토큰의 발급, 갱신, 교환에 대한 방법을 정의한다. WS-Federation은 서로 다른 보안 영역을 통합하기 위한 표준이다.



Reliable Messaging 표준은 application reliability를 개선하기 위한 것이다. 이를 통해 Message는 한접 순서로 배달이 되는 것을 보장한다. 이러한 메시지의 배달은 특정 프로토콜에 국한 된 것이 아니고 HTTP에서도 가능하게 된다. End-to-End 사이의 Message를 신뢰성 있게 전달 할 수 있고 Hop에 관계없이 메시지를 전달 할 수 있다. 메시지는 SOAP 기반의 전달이기 때문에 IP Packet과는 관계가 없다.



Transaction 에 관련 된 표준은 서로 상호 같이 운용이 되는데 WS-Coordination은 Context를 정의하고 참여하는 방법에 관한 정의하고 WS-AtomicTransaction은 Webservice 상의 Two Phase Commit을 지원하며 WS-BusinessActivity는 오래 실행되는 Transaction 에 관한 방법을 정의하는 표준이다.

**서비스 지향**

서비스 지향(Serviced Oriented Architecture)는 특정 기술에 국한된 개념이라기보다는 비즈니스 측면과 기술적인 측면 모두 사용되는 개념이 되었다. 제안 된지는 오래 되었지만 2003년 혹은 2004년에 주목받기 시작한 개념이다.

다음은 SOA에 대한 저명한 정의들이다.

“SOA는 상호 작동하는 시스템 사이를 느슨하게 연결하려는 목적을 가진 아키텍처”

-W3C

“Service-Oriented Architecture(SOA(서비스 지향 아키텍쳐))는 정의가 잘된 인터페이스와 서비스들 간 콘트랙트(contracts)를 통해, 서비스라고 하는 애플리케이션의 다양한 기능 단위를 상호 연관시키는 컴포넌트 모델이다. 인터페이스는 하드웨어 플랫폼, 운영 체계, 프로그래밍 언어에 독립적인 방식으로 정의된다. 따라서 다양한 시스템들에 구현된 어떤 서비스라도 일반적이고 통합된 방식으로 인터랙팅 할 수 있다.” -IBM

"SOA는 공개, 기민성, 확장, 연합, 자립적 요소들로 구성된 조합가능한 아키텍처, 서비스 품질, 다양한 벤더, 상호 운영성, 서비스 발견 그리고 잠재적으로 재사용 가능한 서비스들이 웹서비스로 구현된다. SOA는 비지니스 로직과 기술을 추상화하여, 이 도메인 간에 느슨한 결합을 유도한다. SOA는 과거 플랫폼의 진화물로서, 전통적인 아키텍처의 특징들을 고스란히 가지고 있으며, 명확한 원칙을 가지고 SOE를 지원하며 서비스 지향을 촉진한다. SOA는 엔터프라이즈 환경을 이상적으로는 표준화하지만, 치밀한 사전 계획에 의한 이전 필요성과 현재도 진화하고 있는 기술에 대한 지원만이 이러한 목적을 달성할 수 있다" -Thomas Erl

Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains -OASIS

SOA란 서비비스라 불리는 decoposition된 애플리케이션 조각을 느슨하게 연결해 하나의 완성된 기업어플리케이션을 만드는 아키텍츠이다.

-가트너

위의 정의와 더불어 다시 한번 생각 해 보야 할 부분이 SOA의 필요성 혹은 SOA가 추구하는 목적일 것이다. SOA의 목적은 재사용성과 민첩성이다. 급격히 변하는 비즈니스에 대처 할 수 있는 IT 와 비즈니스 아키텍츠를 구성하는 것이 목적인 것이다. SOA는 시시각각 변하는 비즈니스 프로세스에 빠르게 대응하고 있는 자원을 재활용하고자하는 것인데 SOA는 이로 인해 비즈니스 프로세스에 관심을 둘 수 밖에 없다. BPM 관련 기술과 제품이 프로세스를 관리 했지만 SOA는 보다 넓은 비즈니스 영역에서의 통합과 재활용 민첩성을 목표로 한다.

SOA를 구성하는 요소는 크게 Service 와 Message 이다.

Sercvice는 명확한 기능적인 의미를 지닌 소프트웨어 컴포넌트로, 고차원의 비즈니스 개념을 캡슐화 하고 있는 것을 말한다. 서비스는 자신이 가진 비즈니스 프로세스 캡슐화한 개념이다. SOA 관점에서 본다면 비즈니스 프로세스를 처리할 수 있는 비즈니스 컴포넌트 인 것이다. 서비스는 명확한 인터페이스를 가지고 구현부분과 구별 된다.



서비스는 특정 플랫폼에 독립적이고 동적으로 검색 되고 호출 되어야 하며 자신의 상태를 스스로 유지해야 한다. 위의 그림에서 볼 수 있듯이 Object, Component 보다는 개념상 입자성이 크고 특정 기술에 종속적이지 않다. Component는 플랫폼 종속적인 특징을 가지고 있고 Object는 Component 안에 위치한 보다 작은 입자이다.

서비스는 동적 발견이 가능하고 서비스 제공자와 소비자 사이에는 느슨한 결합이 유지되고 Message를 통하여 통신을 한다. SOA 관점에서 메시지는 특정 기술에 종속적이지 않는 것이다. XML에 기반한 SOAP 메시지가 웹서비스의 메시지로 많이 사용 되는 이유도 여기에 있다.

**SOA 와 WCF**

WCF는 SOA 어플리케이션을 구현하기 위한 Framework이다. 개발자가 SOA 어플리케이션을 WCF를 통해 구현 한다면 SOA를 구현하기 위해 일반적으로 필요한 기술들을 직접 사용하지 않아도 WCF 프레임워크의 도움으로 구현이 가능 한 것이다. 예를 들면 개발자는 WSDL에 대해 자세히 알지 않아도 메시지를 다룰 수가 있다. WCF 프레임워크이 메시지와 WSDL 등을 해석하고 변환하고 참조가 가능하도록 해 개발자는 어플리케이션 로직이나 비즈니스 프로세스에 집중 할 수 있다. 또한 WCF Runtime은 Functionality를 관리한다.

서비스의 Operation 해당하는 부분을 WCF는 Interface로 정의하고 Message를 DataType으로 정의한다. 하지만 개발자는 Language 수준에서만의 혹은 Framework 수준만의 이해가 있다면 특정 기술에 독립적인 프로세스를 노출 할 수 있다. 앞에서 Webservice 표준과 SOA의 개념을 설명한 것은 WCF가 SOA와 Webservice를 어떻게 수용하고 있는가를 이해하기 위함이다.

**WCF Architecture**

다음은 WCF의 Architecture를 그린 다이어그램이다.



Contract는 메시지의 다양한 측면을 정의한다. Data Contract는 Service가 사용 할 각종 Parameter를 설명한다. apt;w; 매개 변수는 XSD Document로 정의 된다.Message contract는 SOAP 프로토콜을 이용해 정의되고 상호 운용성을 높이기 위해 세밀한 부분까지 정의 한다. Service Contract는 실제 사용 할 메서드 명이 되고 C# , Visual Basic 등 언어 수준에서 구현 된 Interface로 배포 된다.

정책과 바인딩은 서비스와의 통신을 위해 필요한 조건을 규정한다. 예를 들면 바인딩에서는 최소한 HTTP 또는 TCP 등의 사용되는 전송 방식과 인코딩을 지정해야 하고 정책에는 보안 요구 사항과 서비스와의 통신을 위해 충족시켜야 하는 다른 조건이 포함된다.

Srevice Runtime 계층에서는 실제 서비스를 수행하는 동안에만 발생하는 동작을 제어한다. Throtting Behavior는 처리되는 메시지의 수를 제어하며, 이는 서비스 요청이 미리 설정한 제한까지 증가한 경우 달라질 수 있다. Error 동작에서는 클라이언트에 전달되는 정보를 제어하는 등의 방법으로 서비스에 내부 오류가 발생하는 경우 일어나는 일을 지정한다. Metadata 동작에서는 메타데이터를 만들어 외부 세계에서 사용할 수 있게 제공하는 방법을 결정한다. 인스턴스 동작에서는 서비스에서 실행할 수 있는 인스턴스의 수를 지정한다. 예를 들어 단일을 지정하면 한 인스턴스에서만 모든 메시지를 처리할 수 있다. 트랜잭션 동작을 사용하면 실패가 발생하는 경우 트랜잭션된 작업을 복원할 수 있다. 디스패치 동작에서는 WCF 인프라로 메시지를 처리하는 방법을 제어한다.

확장성을 사용하면 런타임 프로세스를 사용자 지정할 수 있다. 예를 들어 메시지 검사에서는 메시지 부분을 검사하고, 매개 변수 필터링에서는 메시지 헤더에서 작동하는 필터에 따라 미리 설정된 작업을 발생시킨다.

**Messaging**

메시징 계층은 채널로 구성되고 채널은 메시지를 인증하는 등의 방법으로 메시지를 처리하는 구성 요소이다. 채널 세트를 채널 스택이라고도 한다. 채널은 메시지와 메시지 헤더에서 작동하는데 주로 메시지 본문 내용 처리를 담당하는 서비스 런타임 계층과는 다fms 면이다.

채널에는 전송 채널과 프로토콜 채널의 두 가지 유형이 있다.

전송 채널에서는 네트워크나 기타 외부 세계의 통신 지점으로부터 메시지를 읽고 쓴다. 일부 전송에서는 인코더를 사용하여 XML Infoset으로 표현되는 메시지에서 네트워크에 사용되는 바이트 스트림 표현으로 변환 및 변환 해제한다. 전송의 예에는 HTTP, 명명된 파이프, TCP 및 MSMQ가 있다. 인코딩의 예에는 XML과 최적화된 이진이 있다.

프로토콜 채널은 흔히 메시지의 추가 헤더를 읽거나 써서 메시지 처리 프로토콜을 구현한다. 그러한 프로토콜의 예로는 WS-Security와 WS-Reliability가 있다.

메시징 계층에서는 데이터의 가능한 형식과 교환 패턴을 나타낸다. WS-Security는 메시지 계층에서 보안을 활성화하는 WS-Security 사양을 구현한 것이다. WS-Reliable 메시징 채널을 사용하면 메시지 배달을 보장할 수 있다. 인코더에서는 메시지의 필요에 맞게 사용할 수 있는 다양한 인코딩을 제공한다. HTTP 채널에서는 메시지 배달에 HTTP(HyperText Transport Protocol)를 사용하도록 지정한다. TCP 채널에서는 마찬가지로 TCP 프로토콜을 지정한다. 트랜잭션 흐름 채널은 트랜잭션되는 메시지 패턴을 결정한다. 명명된 파이프 채널을 사용하면 프로세스 간 통신이 가능하다. MSMQ 채널을 사용하면 MSMQ 응용 프로그램과 상호 작용할 수 있다.

**Activating And Hosting**

자체 실행 파일이 서비스를 호스팅 할 수 있는데 이를 자체 호스팅이라고 한다.

서비스를 호스팅하여 IIS 또는 WAS(Windows Activation Service)와 같이 외부 에이전트에서 관리되는 실행 파일로 실행할 수도 있다. WAS를 사용하면 WAS를 실행하는 컴퓨터에 배포된 WCF 응용 프로그램을 자동으로 활성화할 수 있고 실행 파일(.exe 파일)을 통해 서비스를 수동으로 실행할 수도 있다. 또 서비스를 Windows 서비스로 자동 실행할 수도 있고 COM+ 구성 요소를 WCF 서비스로 호스팅할 수도 있다.

**Message Sending**

WCF에서 Service는 Endpoint를 통해 Message를 수신한다. 하나의 Service에 대해 여러 개의 Endpoint를 노출한다. Client는 Proxy를 통해 메시지를 주고받는다. Proxy가 내부적으로 투명하게 작동함으로서 통신상의 작용하는 복잡한 Chanel 등에 대한 이해 없이도 사용 할 수 있다.



WCF Dispatcher는 전달된 메시지를 변환 해 Service 객체의 메소드 호출 요청으로 변환한다.