WCF som lösning för kommunikation mellan applikationer för 32- respektive 64-bitars ordlängd

Niklas Bäckström

Examensarbete

Informationsteknik

2013

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	4171
Författare:	Niklas Bäckström
Arbetets namn:	WCF som lösning för kommunikation mellan applikationer för 32- respektive 64-bitars ordlängd
Handledare (Arcada):	Hanne Karlsson
Uppdragsgivare:	

Sammandrag:
Avsikten med detta examensarbete är att utreda hur Windows Communcation Foundation(WCF) passar för att lösa problemet med att använda 32-bitars komponenter i 64-bitars applikationer. Problemet är att 64-bitars applikationer inte direkt kan använda sig av 32-bitars komponenter utan det krävs en applikation eller tjänst som förmedlar kommunikationen. Examensarbetet är uppdelat i en teoretisk och en praktisk del. Den teoretiska delen berättar vad tjänster och WCF är och hur de fungerar. Teoretiska delen går även in på hur en WCF tjänst och klient kan konfigureras. I den praktiska delen visas hur kommunikation mellan processer på en dator kan implementeras med hjälp av WCF. Den praktiska delen är programmerad i C# och i märkningspråket XML. Slutsatsen av examensarbetet är att WCF mycket väl kan användas som en metod för interprocesskommunikation mellan två komponenter på en dator.

Nyckelord:	Windows Communcation Foundation, WCF, IPC, .NET, Service, SOA
Sidantal:	40
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information technology
Identification number:	4171
Author:	Niklas Bäckström
Title:	WCF as a solution for communication between 32 - and 64-bit applications
Supervisor (Arcada):	Hanne Karlsson
Commissioned by:	

Abstract:
The purpose of this thesis is to explore how Windows Communication Foundation(WCF) suits to solve the problem of using 32-bit components in 64-bit applications. The problem is that 64-bit applications can't directly make use of 32-bit components without an application or service that mediates the communication. The thesis is divided into a theoretical and a practical part. The theoretical selection describes what services and WCF are and how they work. The theoretical part also shows how a WCF service and client can be configured. The practical part demonstrates the implementation of inter-process communication using WCF. The practical example is programmed in C# and the markup language XML. The conclusion of the thesis is that WCF is well suited for inter-process communication between two components within a computer.

Keywords:	Windows Communcation Foundation, WCF, IPC, .NET, Service, SOA
Number of pages:	40
Language:	Swedish
Date of acceptance:	

INNEHÅLL

Figurer

Tabeller

Förkortningar

CLI	Common Language Infrastructure
CLR	Common Language Runtime
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
DLL	Dynamic-link Library
GUI	Graphical User Interface
IIS	Internet Information Services
IPC	Inter-process Communication
MSMQ	Microsoft Message Queuing
OLE	Object Linking and Embedding
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
WCF	Windows Communication Foundation

1 INLEDNING

1.1 Bakgrund

I dagens läge sköts en stor del av kommunikationen mellan klient / värd applikationer med hjälp av olika tjänster. Det finns många olika sätt att implementera en tjänst. Detta examensarbete handlar om att noggrannare utforska i WCF (Windows Communication Foundation), som används för att bygga tjänstebaserade applikationer. Med hjälp av WCF går det att utveckla totalt generiska klient/värd applikationer som stöder olika W3C standarder. Det antas att läsaren är bekant med programmeringsspråket C# och märkningsspråket XML.

1.2 Syfte och målsättning

Syfte med detta arbete är att skapa sig en klar bild över hur det lönar sig att utveckla och designa WCF tjänster; att försöka beskriva de vanligaste problemen och undvika fallgroparna som kommer fram under utvecklingen. Syftet är även att presentera ramverket och dess funktioner på ett strukturerat sätt med tydliga exempel som hjälper läsaren att förstå hur de olika delar skall konfigureras. Eftersom WCF ramverket har blivit så omfattande behandlar examensarbetet hur användaren skall få ut den önskade funktionaliteten ur ramverket.

Målet är även att få en klar översikt på vad som ligger under WCF d.v.s. vad den tjänsteorienterade arkitekturen innebär och på vilka sätt man kan implementera paradigmen från tjänsteorienterade arkitekturen med hjälp av WCF.

Hur skall man gå till väga för att utnyttja 32-bitars program eller programbibliotek i 64-bitars applikationer? I examensarbetet demonstreras en lösning till detta problemet med hjälp av WCF.

1.3 Avgränsningar

WCF tjänster måste alltid ha en värd för att kunna användas. Det finns många olika sätt att vara en värd, men i detta examensarbete behandlas inte tekniken bakom de olika sätten att vara värd för en WCF tjänst.

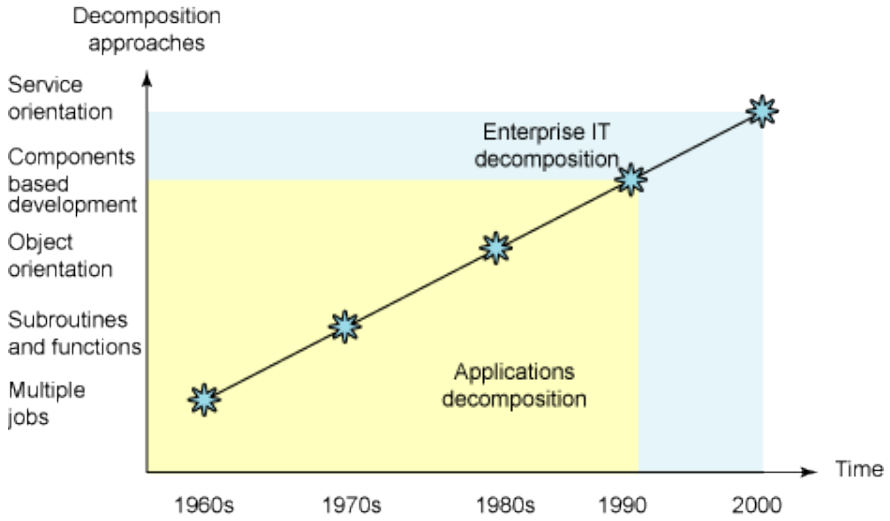
Riktiga exempel eller vilka applikationer det handlar om kan på grund av krav på sekretess inte användas i examensarbetet.

2 TJÄNSTEORIENTERAD ARKTIEKTUR

Tjänsteorienterad arkitektur (eng. Service Oriented Architecture, härefter SOA) är en uppsättning principer och metoder för att designa och utveckla tjänster.

2.1 Bakgrund

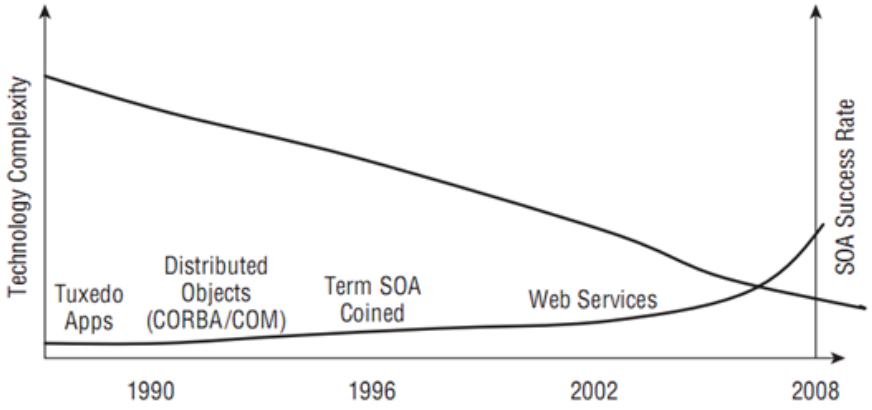
Enligt systemteorin är systemets storlek direkt bundet till hur många okända delar systemet innehåller; ju större system desto mera okända aspekter. För att lösa detta har uppdelning av system i mindre, mera hanterbara, delar redan utnyttjats sedan 1960-talet. Figur 1 demonstrerar olika teknologierna som finns för att lösa problemet med uppdelningen av system. En del av de tidigare metoderna, så som objektorientering och komponentbaserad utveckling, har lyckats bra med uppdelningen på applikationsnivå. SOA ger en vidare möjlighet att återanvända delar i hela företagets verksamhet och applikationer, detta har fört uppdelning av system till en högre nivå. (Lublinsky, 2007)



Figur 1 Evolution av metoder för uppdelning. (Lublinsky, 2007)

SOA har redan länge existerat och definitionen av SOA uppkom första gången som skriven text år 1996 i en rapport utgiven av Gartner(Josuttis 2007 s. 7). Redan innan det hade stora finans- och telekommunikationsföretag börjat använda sig av arkitektur som liknade SOA. Teknologier som företagen använde var olika slags distribuerade tekniker så som CORBA och DCOM. I Figur 2 illustreras en tidslinje där det framkommer utvecklingen och användningsgraden av SOA. Som figur 2 visar fick SOA kring år

2000 sin största drivkraft, då Microsoft lanserade den första webbtjänsten Web Services. Web Services underlättade utveckling av SOA baserade tjänster på ett märkbart sätt. I dagens läge ser många ordet webbtjänst som synonym till SOA även om SOA är mycket mera än bara webbtjänster.



Figur 2 SOA tidslinje (Rosen et al. 2008 s. 7)

Enligt Rosen et al. (2008 s. 8) misslyckades en stor en stor del av de första SOA genomföranden. Två olika orsaker framträdde när man noggrannare granskade misslyckanden. En av svårigheterna var att teknologierna som användes för att genomföra SOA var för komplexa för en medelmåttig programmerare. Det andra som framkom var att ingen ännu visste vad en bra tjänst innebar. Det som historien har lärt oss är att lyckade SOA genomföranden inte är baserade på teknologierna. Ett lyckat SOA-genomförande kräver en investering i arkitektur, tid, affärs- och strategisk vision, ingenjörsarbete och förvaltning.

2.2 Definition

Det förekommer inte en gemensam konsensus på den exakta betydelsen av SOA. Personer i olika positioner ser SOA på olika sätt. Ur ett affärsperspektiv är SOA en uppsättning tjänster eller dvs. IT-tillgångar vilka i sin tur kan användas för att bygga lösningar för kunder och partners. En projektledare ser SOA som en utvecklingsmodell som ger möjlighet till en parallell utveckling. För en programutvecklare är SOA ett ramverk med verktyg och teknologier, t.ex. webbtjänster. (Lublinsky, 2007)

SOA experter är överens om att SOA är ett paradigm för en förbättrad flexibilitet. SOA är ett paradigm för att det inte finns ett ramverk eller en färdig produkt utan SOA är någonting som leder till en arkitektur. Mer specifikt sagt kan man med SOA bygga fristående affärstjänster som kan kombineras till en betydelsefull högnivå process och lösning. (Rosen et al. 2008 s. 33) (Josuttis 2007 s. 12)

2.3 Motivering

Ett stort företag har ofta många olika datasystem för att de skall kunna driva sin verksamhet. Systemen bör ha en hög integrationsgrad och det sker konstant ändringar i systemen. SOA ger en möjlighet att enkelt sköta kommunikationen mellan olika system på ett teknikneutralt sätt. SOA minskar problemen som uppstår med stora IT-system och därför är paradigmet ypperligt vid lösning av dessa problem. (Josuttis, 2007) (OASIS Reference model, 2006)

OASIS Reference model (2006) definierar att distribuerade system ofta styrs av olika enheter eller t.o.m. olika företag. Dessa nätverk av distribuerade system kan ha olika egenskaper t.ex. olika budgeter och prioriteter. När det blir större och fler system som skall knytas ihop, blir SOAs totala neutralitet till plattform och teknik en bra orsak att implementera SOA.

Ofta försöker företag förenhetliga sina system för att göra underhåll och integration enklare. Detta är dock i praktiken omöjligt så fort det handlar om ett större system. SOA ser ingen konflikt i att system är heterogena utan stöder och godkänner denna egenskap på samma sätt som viga utvecklingsmetoder accepterar att det sker ändringar i kraven.

2.4 Tekniska koncept

2.4.1 Tjänster

En tjänst i IT världen definieras som en fristående och tillståndslös affärsfunktion som tar emot en eller flera förfrågningar och ger ett svar dessa förfrågningar genom ett väldefinierat och standardiserat gränssnitt. Dessa affärsfunktioner kan t.ex. vara "Ny kund" eller "Skicka meddelande". I detta sammanhang betyder fristående att tjänsten skall ha så lite beroenden till olika delar i ett system som möjligt. Vissa beroenden kommer dock alltid att finnas t.ex. fundamentala datatyper så som string eller heltal. Tillståndslös betyder att tidigare händelser inte påverkar förfrågningar och att funktionen utförs på samma sätt varje gång.

Tjänster kan utföra arbete så som modifiering och bearbetning av data. Funktionaliteten i en tjänst definieras via ett service gränssnitt, som kan vara uppbyggd av många olika byggstenar. Detta innebär att en tjänst inte är bunden till en viss teknologi utan är snarare en arkitektonisk artefakt som utnyttjas i genomförandet av företagets affärslösningar. (Josuttis, 2007) (Lublinsky, 2007)

2.4.2 Lös koppling

I databehandling och systemdesign betyder lös koppling (eng. Loose Coupling) ett system där varje komponent inte har stor, eller någon kunskap om de övriga definitioner. Lös koppling är ett koncept där beroenden mellan systemen minimeras så att även om en del blir funktionsoduglig så fortsätter systemet att funktionera.

Övriga positiva aspekter med lös koppling är att det leder till stor skalbarhet. I stora system bildas ofta s.k. flaskhalsar som fördröjer utförandet av tjänsten eller processen. Lös koppling betyder att man strävar till att bygga systemet så decentraliserat som möjligt vilket i sin del hjälper med att förhindra bildningen av flaskhalsar i centraliserade funktioner. Mängden av lös koppling varierar enligt systemets och

verksamhetens behov. I tabell 1 visas olika former av lös koppling som används inom SOA. (Josuttis, 2007)

Tabell 1 Olika former av lös koppling

	Tät koppling	Lös koppling
Förbindelsen	Punkt-till-punkt	Via förmedlare
Kommunikation	Synkron	Asynkron
Datamodell	Komplexa typer	Simpla typer
Kontroll av logik	Centraliserat	Decentraliserat
Driftsättning	Samtidig	Vid olika tidpunkter

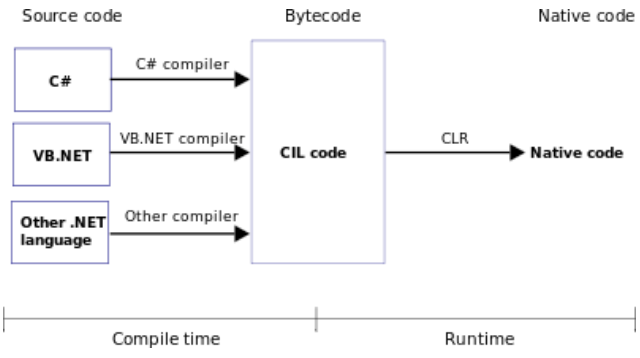
3 WCF

Windows Communication Foundation (WCF) är en körtidsmiljö och ett programmeringsgränssnitt i .NET Ramverket som används för att bygga serviceorienterade applikationer.

3.1 .NET Ramverk

.NET ramverket är ett programmeringsramverk utvecklat av Microsoft. Ramverket består av ett stort klassbibliotek som innehåller vältestade och återanvändbara kodbibliotek. Utvecklare kan utnyttja dessa kodbibliotek i sina egna applikationer. Det finns ett antal olika programmeringsspråk som kan användas i .NET. Det populäraste språket, som även används som exempel i detta examensarbete, är c#. Oberoende av programmeringsspråk kompileras all källkod i .NET applikationer till Common Language Infrastructure (CLI) kod. CLI är den lägsta nivån av kod i ramverket. Nyttan med att kompilera allt till CLI kod är att det går att utnyttja alla kodbibliotek som är skrivna i ett av flera .NET programmeringsspråken.

Det andra stora elementet i ramverket är Common Language Runtime (CLR) som är ansvarig för exekveringen av hanterad .NET kod. CLR är den delen som omvandlar CLI till nativ kod som datorns processor sedan exekverar. CLR ger medvärde till applikationer i form av minneshantering, trådhantering och felhantering. I figur 3 illustreras flödet från källkod till nativkod. (Microsoft, 2012b)



Figur 3 Visuell överblick på CLI och CLR (Wikipedia, 2013a)

Applikationer som exekveras på CLR har möjlighet att använda sig av alla funktioner som kommer med CLR. Dessa applikationer kallas hanterade applikationer(eng. managed code). Den andra typen av applikationer är icke hanterade applikationer som inte körs på CLR utan nativt på operativsystemet. En viktig egenskap i ramverket är att icke hanterade applikationer kan även inuti sig ladda upp CLR i sin process och exekvera hanterad kod. Ett exempel på detta är Internet Explorer som har utvidgningar som är utvecklade som hanterade applikationer. Hanterade applikationer kan även ladda icke hanterad kod i sig för att köra mera prestandakritiska funktioner. (Wikipedia, 2013a)

.NET används inte enbart för att utveckla PC- och webbapplikationer utan en avskalad version av ramverket stöder även mobiltelefoner och inbyggda system. Microsoft erbjuder även en integrerad utvecklingsmiljö som heter Visual Studio. Nyaste versionen för .NET är 4.5 och för Visual Studio 2012. (Microsoft, 2012b)

Dessa versioner används i alla praktiska exempel i detta examensarbete.

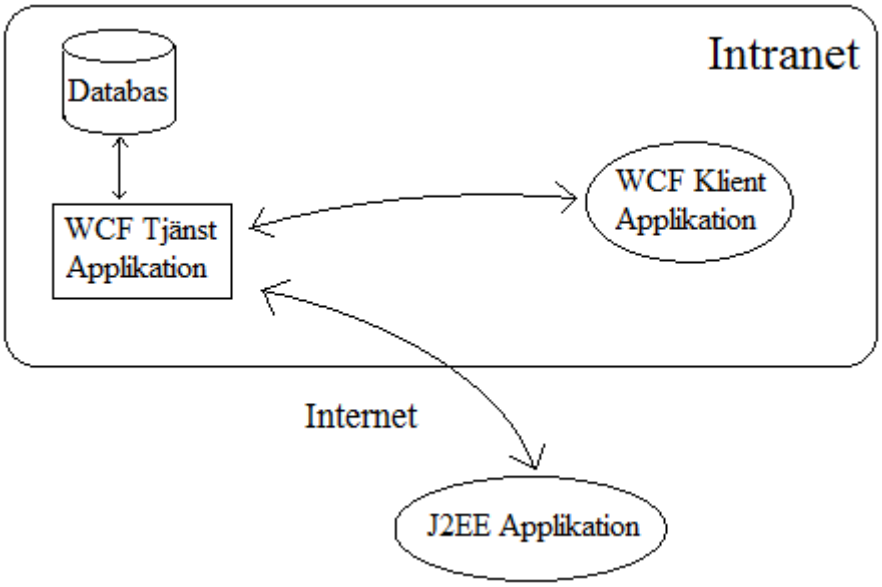
3.2 Inledning till WCF

WCF är Microsofts lösning för att förverkliga en uppsättning standarder; t.ex. interaktion mellan tjänster, typomvandling och hantering av olika protokoll. Därmed ger WCF interoperabilitet mellan tjänster. Med hjälp av WCF kan meddelanden skickas asynkront från en tjänsteslutpunkt till en annan slutpunkt. Meddelandet kan variera från simpla datatyper, som ett ord, till mer komplex format, som binär data eller multidimensionella listor. WCF meddelanden är oftast inkapslade i ett SOAP meddelande men .NET version 4.0 stöder WCF även i andra format så som Representational State Transfer. (REST) (Microsoft, 2012a)

Tjänster byggda med WCF ramverket kan både kommunicera med klienter som är byggda med WCF och klienter som är byggda med någon annan teknologi. Samma gäller för klienter, WCF klienter kan kommunicera med tjänster som inte är byggda med WCF.

Fördelen om både tjänsten och klienten är skapade med WCF är självklar. Det går att utnyttja egenskaper som är färdigt inbyggda i WCF, t.ex. automatiskt generering av klientkod.

I figur 4 nedan demonstreras hur ett tjänstebaserat system kan vara uppbyggt. I figuren visas en WCF tjänst som tillhandahåller en funktion att hämta information från en databas. En WCF klient och en extern applikation kan båda anropa WCF tjänsten. WCF tjänsten ger samma respons till klienterna oberoende av tekniken som används i klienterna. (Löwy, 2010)



Figur 4 Exempel på ett tjänstebaserat system.

Första versionen av WCF lanserades med .NET 3.0 och den nuvarande versionen lanserades med .NET 4.5. Största förbättringarna har gjorts i förenklingen av konfiguration och underhåll för en WCF tjänst. Tekniskt sätt fungerar ramverket ungefär på samma sätt som i tidigare versioner. (Microsoft, 2012a)

3.3 Slutpunkter

Att kommunicera med WCF tjänster sker alltid mellan slutpunkter. Slutpunkten ger klienterna tillgång till tjänstens funktionalitet. En slutpunkt består av följande fyra delar:

- En adress som berättar var slutpunkten ligger.
- En bindning som specificerar hur klienten kan kommunicera med slutpunkten
- Ett kontrakt som visar vilka operationer tjänsten innehåller
- En uppsättning beteenden som anger detaljer av implementeringen.

(Microsoft, 2012a)

3.3.1 Adresser

En slutpunktadress används för att känna igen var i nätverket tjänsten finns, vilket kan jämföras med en fysisk gatuadress. Dessutom kan adressen innehålla en identifierande egenskap som möjliggör att andra slutpunkter kan autentisera slutpunkten. En WCF adress har följande struktur:

[transportprotokoll] :// [Maskin] [:(tillval)Port] / [Sökväg]
Exempel på en adress: http://localhost:8000/MinTjanst/slutPunkt

Slutpunktadressen kan antingen programmeras direkt i kod eller konfigureras med hjälp av en XML konfigurationsfil. Konfigurering via XML rekommenderas, eftersom det då går att göra ändringar i adressen utan att bli tvungen att kompilera och distribuera applikationen på nytt. Det finns två olika sätt att specificera adressen för en slutpunkt. Ena sättet är att definiera en basadress för hela applikationen och sedan definiera relativa adresser för varje slutpunkt. Det andra sättet är att definiera absoluta adresser för varje slutpunkt. (Microsoft, 2012a)

Konfigurationen av adresser visas senare i kapitel 5.

3.3.2 Bindningar

Bindningar är objekt som specificerar de kommunikationsdetaljer som krävs för att kunna ansluta sig till en slutpunk. En bindning är uppbyggd av tre olika element:

- Protokollelement som definierar säkerhet och pålitlighet i slutpunkten.
- Transportelement som definierar det underliggande transportprotokollet t.ex. TCP eller HTTP
- Meddelandeelement som definierar vilken kodning skall användas med meddelandet som skickats till slutpunkten t.ex. text eller binär.

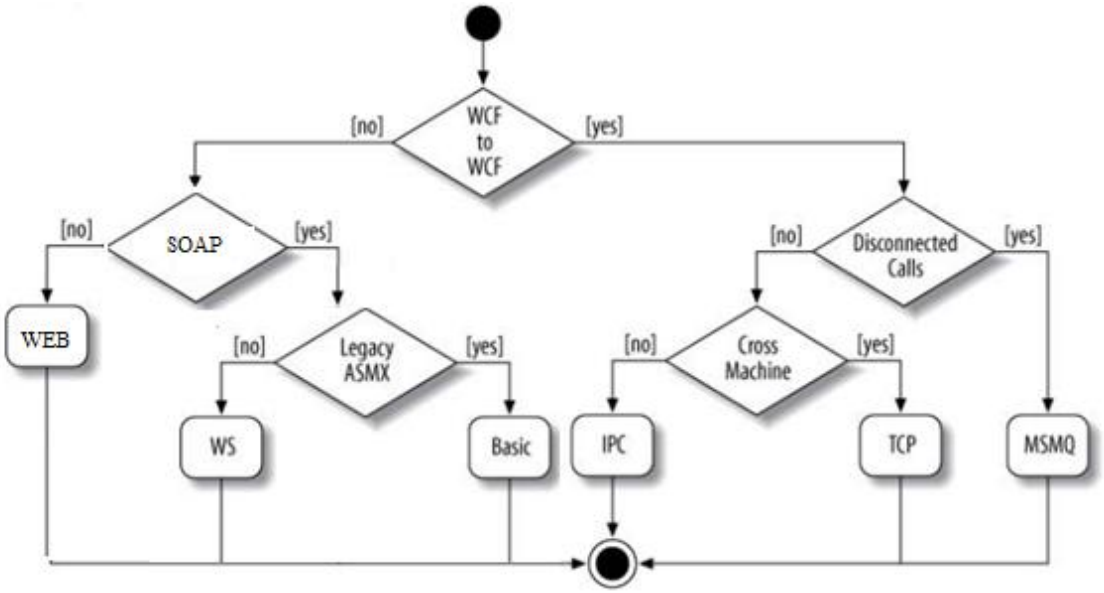
WCF ger en möjlighet att bygga anpassade bindningar. Problemet med skräddarskydda bindningar är att det finns tusentals(Löwy, 2010. s24) möjliga slutresultat av en kombination av olika element. En stor del av kombinationerna fungerar inte på grund av att vissa val av element utesluter andra element eller kräver andra element för att fungera. För att underlätta användningen av bindningar har WCF färdigt inbyggt en stor mängd fördefinierade bindningar som kan användas av största delen av scenarier. De mest använda och nyttigaste bindningar visas i tabellen nedan:

Tabell 2 Populära bindningar (Microsoft, 2012a)

Binding	Description
BasicHttpBinding	A binding that is suitable for communicating with WS-Basic Profile conformant Web services, for example, ASP.NET Web services (ASMX)-based services. This binding uses HTTP as the transport and text/XML as the default message encoding.
WSHttpBinding	A secure and interoperable binding that is suitable for non-duplex service contracts.
WSDualHttpBinding	A secure and interoperable binding that is suitable for duplex service contracts or communication through SOAP intermediaries.
NetTcpBinding	A secure and optimized binding suitable for cross-machine communication between WCF applications.

NetNamedPipeBinding	A secure, reliable, optimized binding that is suitable for on-machine communication between WCF applications.
NetMsmqBinding	A queued binding that is suitable for cross-machine communication between WCF applications.
WebHttpBinding	A binding used to configure endpoints for WCF Web services that are exposed through HTTP requests instead of SOAP messages.

Med hjälp av figur 5 går det att välja rätt bindning för olika scenarier. (Microsoft, 2012a)



Figur 5 Val av bindning

Första frågan vid val av bindning är om kommunikationen sker mellan två WCF slutpunkter. Om det handlar om WCF till WCF kommunikation som inte kräver ansluten kommunikation är det MSMQ som rekommenderas. Om klienten är konstant ansluten till tjänsten och inte ligger på samma dator som tjänsten så rekommenderas TCP. Om kommunikationen sker på samma dator lönar det sig att välja IPC som har den bästa prestandan.

Ifall det inte handlar om WCF till WCF kommunikation är nästa fråga om det skall handla om en SOAP tjänst. Om det inte är SOAP tjänst finns det bara WEB som kan

användas. Ifall det handlar om SOAP används Basic med legacy-system och WS ifall det är moderna webbtjänster.

Om ingen av de färdigt definierade bindningarna är passande så går det att modifiera dem för att få en bindning som passar det egna scenariot. (Löwy, 2010)

3.3.3 Kontrakt

Alla WCF tjänster publicerar kontrakt som beskriver den funktionalitet som slutpunkten erbjuder. Ett kontrakt är ett plattforms oberoende sätt att beskriva vad tjänsten kan utföra. Servicekontrakt räknar upp olika operationer som kan anropas av klienten. Servicekontrakten innehåller även definitionen om i hurudan form meddelandet är, vilka input parametrar krävs för att anropa operationen och hurudant svarsmeddelande klienten kan vänta sig. (Microsoft, 2012a)

I kapitel 5 behandlas hur kontrakten ser ut och hur de implementeras.

3.4 Utbyte av Metadata

Metadata, som betyder information om data, används i WCF för att beskriva hur man interagerar med tjänstens slutpunkter. En klient kan antingen få kunskapen för en slutpunkt genom en publicering av metadata eller genom att importera metadata från en fil så som ett klassbibliotek.

Publicering av metadata gör metadata tillgänglig över standard protokollet MEX (Metadata Exchange Endpoint) eller HTTP/GET. Metadata slutpunkter ser ut så som normala slutpunkter. De har en adress, en bindning och ett kontrakt. Kontraktet för MEX slutpunkter är alltid "IMetdataExchange".(Löwy, 2010)

I kapitel 5 behandlas hur utbyte av metadata implementeras.

3.5 Säkerhet

Säkerheten i WCF kan delas upp i tre olika huvuddelar: Autentisering av klienten, säkerheten vid överförningen av meddelandet och säkerheten av själva meddelandet. Speciellt noggrann säkerhetsdesign måste tas i beaktande då data skall överföras över ett öppet nätverk. Säkerheten i WCF bygger på koncept och teknologier som redan används och utnyttjas i andra säkerhetslösningar, exempelvis SSL och HTTPS. (Microsoft, 2012a)

3.5.1 Autentisering

Autentisering betyder att kunna verifiera att anroparen av en tjänst faktiskt är den som anroparen påstår sig vara. Autentisering handlar även om att kunna säkerställa att tjänsten som en klient har anropat faktiskt är den rätta och ingen har kapat klientens anrop och omdirigerat anropet. Olika metoder av autentisering i WCF är bl.a.

- Windows autentisering med Kerberos.
- Användarnamn och lösenord.
- Certifikat.
- Skräddarsydda metoder så som biometrisk identifikation.

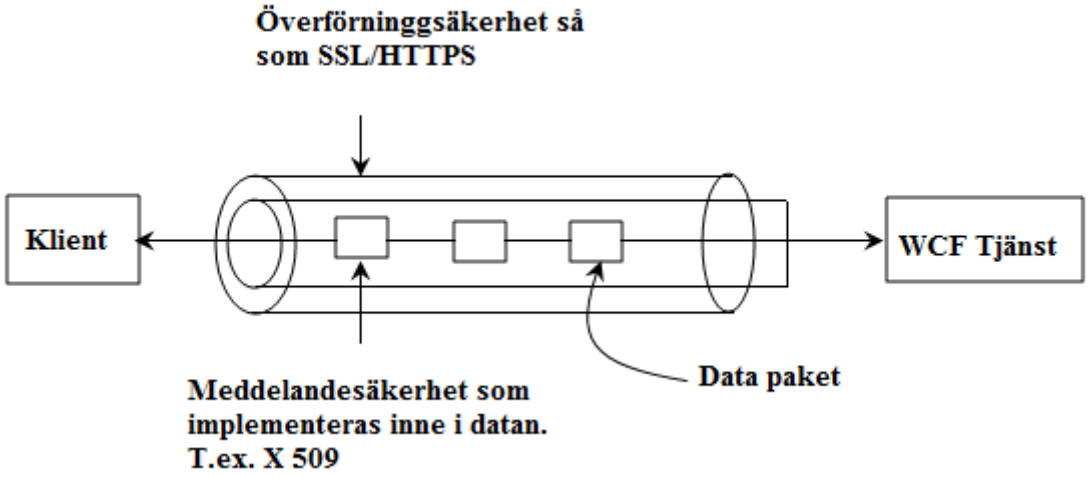
(Microsoft, 2012a)

3.5.2 Överförnings- och meddelandesäkerhet

Autentisering är i sig viktigt, men om överförningen av meddelandet inte är säkert blir autentiseringen irrelevant. Olika aspekter vid överförningssäkerhet är integritet, sekretess och ömsesidig autentisering. Integritet betyder att meddelandet inte har manipulerats på vägen från slutpunkt till slutpunk. Sekretess betyder att ingen annan än den menade mottagaren har läst meddelandet.

WCF stöder fem olika överförnings- och meddelandesäkerhetsmetoder. Överförnings- eller transportsäkerheten krypterar själva kommunikation mellan slutpunkterna.

Message- eller meddelandesäkerheten krypterar meddelandet vilket betyder att meddelandet kan överföras över osäkra kanaler så som HTTP och ändå behålla sin integritet. Vid mixed mode användes meddelandesäkerhet för att kryptera avsändarens autentiseringsuppgifter och överförningssäkerhet för resten av säkerheten. Figur 6 nedan visar hur meddelande- och överförningssäkerhet ser ut.



Figur 6 Visuell blick på meddelande- och överförningssäkerhet

De fem olika WCF överförningssäkerhet metoderna och vilka bindningar som stöder vilka metoder syns i figuren nedan. (Löwy, 2010)

Name	None	Transport	Message	Mixed	Both
BasicHttpBinding	Yes (default)	Yes	Yes	Yes	No
NetTcpBinding	Yes	Yes (default)	Yes	Yes	No
NetNamedPipeBinding	Yes	Yes (default)	No	No	No
WSHttpBinding	Yes	Yes	Yes (default)	Yes	No
NetMsmqBinding	Yes	Yes (default)	Yes	No	Yes

Figur 7 Bindningar och säkerhetsmetoder som de stöder.

Bindningar som används vid intranät tjänster har som förinställt överförningssäkerhet. Överförningssäkerhet används för att den har bra prestanda. Bindningar som används för tjänster över ett öppet nätverk har i sin del förinställt meddelandesäkerhet. (Microsoft, 2012a)

3.6 Vara värd för WCF tjänst

Varje WCF tjänst behöver något som är värd för tjänsten, dvs. en värdprocess på en Windows maskin. En värdprocess kan samtidigt fungera som värd för många olika tjänster. I följande underrubriker beskrivs fem olika metoder för att vara värd för en WCF tjänst.

3.6.1 Anpassat program

Med ett anpassat program menas att programmet som utnyttjar tjänsten fungerar som värd, denna metod kallas på engelska för self-hosting. Alla applikationer byggda med .NET ramverket kan fungera som värd för en WCF tjänst genom att implementera ServiceHost klassen. Att WCF tjänsten är inbyggd i en .NET applikation är det mest flexibla sättet att ta i bruk en tjänst eftersom det kräver minst arbete och konfiguration. Nackdelarna med denna metod är att värdapplikationerna inte automatiskt erbjuder funktionalitet som underlättar hanteringen av tjänster, funktionaliteter så som automatisk start av tjänsten på ett anrop. Detta sätt används ofta i utvecklingsfasen p.g.a. snabb driftsättning. (Jöwy, 2010)

3.6.2 Windows tjänst

Användningen av Windows tjänster som värd passar speciellt bra för applikationer som skall vara igång hela tiden. Med Windows tjänster kan WCF tjänsten automatiskt sättas igång då servern startas till skillnad från att använda IIS där tjänsten aktiveras med första anropet. Användning av en Windows tjänst som värd har även den fördelen att tjänsterna stängs av då man själva vill. Tjänster som har IIS/WAS som värd kan även stänga sig själva automatiskt för att spara på server resurser.

Windows tjänst kan användas som värd för WCF från och med Windows XP och Windows server version 2003. Ingen extra mjukvara behöver installeras på servern för att använda Windows tjänst som värd. Dessa tjänster administreras i Windows via

Services konsol. Windows tjänster installeras med hjälp av Installutil.exe programmet som är inbyggt i .NET. (Microsoft, 2012a)

3.6.3 IIS 5/6

Även om IIS 5/6 är gammalt finns det ännu en hel del av dem i drift. Största fördelen med att använda IIS som värd är att tjänsten automatiskt aktiveras med första anropet och att IIS hanterar livscykeln för värdprocessen. Största nackdelen med IIS 5/6 är att de inte stöder andra protokoll än HTTP/HTTPS. För att använda IIS 5/6 som värd krävs det en .svc fil i tjänste lösningen som definieras på följande sätt:

```
<%@ ServiceHost
    Language="C#"
    Debug="true/false"
    Service="Namnet på tjänsten"
    CodeBehind="Filen där tjänsten ligger"
%>
```

3.6.4 IIS7/WAS

Windows Process Activation Service (WAS) är en värdmotor som bygger sig på IIS. Det går även att installera WAS utan att installera IIS. WAS är en värdmotor för allmänt ändamål som kan utnyttja alla transportprotokoll som WCF använder. Problemet med WAS är att den egentigen inte är optimerad för att vara värd för tjänster d.v.s. den vet inte om det är en websida eller en tjänst som den är värd för. (Microsoft, 2012a)

3.6.5 AppFabric

AppFabric är den nyaste metoden som kan användas som värd for WCF tjänster.
Den största nyttan för WCF är att AppFabric kan fungera på samma sätt som en Windows tjänst, den kan automatiskt starta tjänster utan att behöva vänta på första klientens anrop. AppFabric kom ut med IIS 7.5 och är en utvidgning till WAS. (Löwy, 2010)

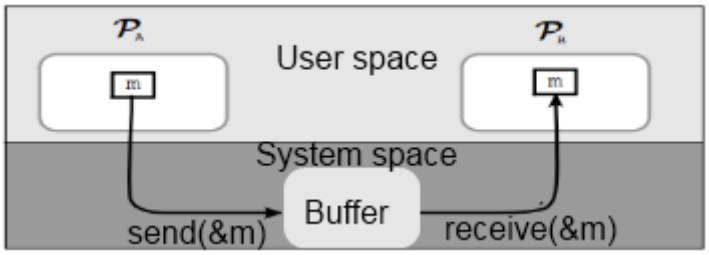
4 KOMMUNIKATION MELLAN PROCESSER

Det som krävs för att kunna utnyttja 32-bitars komponenter i 64-bitars applikationer är någon form av kommunikation mellan processer (Interprocess Communication, IPC). I WCF implementeras IPC inom samma dator med hjälp av NetNamedPipeBinding. Denna bindning används i exemplet i kapitel 5. I detta kapitel berättas generellt vad IPC innebär och mera ingående vilka olika metoder som finns för IPC i operativsystemet Windows.

4.1 Introduktion

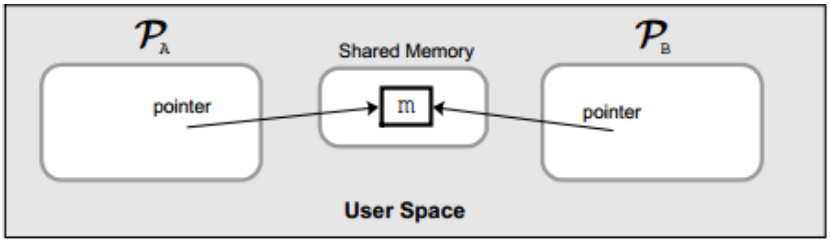
IPC är en uppsättning metoder för utbyte av data mellan flera trådar i en eller flera processorer i en dator eller med andra datorer i nätverket. Det finns många orsaker varför IPC behövs i ett operativsystem. Den främsta orsaken är att data måste kunna transporteras mellan applikationer. En annan stor nytta är återanvändbarhet, flera applikationer kan kalla på en process istället för att ha en egen tillämpning. IPC är även nyttig för säkerheten av applikationer i form av rättighets-separation. Då ett program skall sköta en säkerhetskritisk uppgift kan man separera den biten så att inte hela programmet skall behöva höga rättigheter. Därefter kan kommunikationen mellan de olika delarna skötas med hjälp av IPC. (Wikipedia, 2012a) (Kuhns, 2004)

Det två huvudsakliga metoder för förverkligandet av IPC är meddelanden och delat minne. Meddelanden är nuförtiden det vanligare sättet. Meddelandemetoden består av två delar. Först etableras en kommunikationslänk mellan processerna. Efter det kan utbyte av meddelanden ske genom att skicka och motta meddelanden över den etablerade länken. Figur 8 illustrerar hur meddelande metoden fungerar.



Figur 8 Meddelandemetoden

IPC över delat minne fungerar genom att en process sparar data i RAM som andra processer har möjlighet att använda. Delat minne är snabbare än meddelande metoden. Nackdelarna med delat minne är att det är svårare att genomföra och kommunikationen är begränsad till en dator. Delat minne används även i kommunikation mellan trådar i ett program med flera trådar. I figur 9 visas delat minne metoden. (Kuhns, 2004) (Kubiatowicz, 1998)



Figur 9 Delat minne metoden

4.2 Tillämpningar av IPC i Windows

Alla operativsystem borde tillhandahålla olika sätt att för kommunicera och dela data mellan olika applikationer. Windows innehåller totalt nio olika IPC metoder för detta ändamål. För att välja den rätta metoden måste man kunna svara på följande frågor:

- Bör applikationen kunna kommunicera med applikationer på samma nätverk eller enbart med applikationer på samma dator?
- Bör applikationen kunna kommunicera med applikationer som kör på helt olika operativsystem?
- Bör användaren av applikationen själva välja vilka andra applikationer den skall kommunicera med eller hittar applikationen sina samarbetspartners av sig själva?
- Bör applikationen kunna kommunicera med ett stort antal olika applikationer på ett generellt sätt eller finns det ett begränsat antal interaktioner med andra specifika applikationer?
- Är prestandan en kritisk del i applikationen? Vissa IPC metoder är tyngre än andra.

- Bör applikationen innehålla en GUI? Vissa IPC metoder kräver GUI funktionalitet.

Efter att dessa sex frågor är besvarade kan man välja en av metoderna som uppfyller de egna kraven.

Första metoden är urklipp. Urklipp fungerar som ett central ställe för datadelning mellan applikationer. Urklipps funktionalitet är en relativt självklar IPC metod och fungerar ur lådan då man använder färdiga element i .NET så som TextBox element eller motsvarande. Alla applikationer har möjlighet att söka data från urklipp. (Microsoft, 2011a)

En annan metod i Windows miljön är Object Linking and Embedding (OLE). Denna metod ger möjlighet för applikationer att kalla på andra applikationer för att editera data. Ett bra exempel på OLE är Word, då man editerar t.ex. ett kalkylblad inne i Word är det egentligen Excel som sköter om editeringen. Båda applikationerna måste implementera OLE för att kunna utnyttja denna funktionalitet. (Microsoft, 2011a)

Information kan även överflyttas mellan applikationer med lågnivåmetoden WM_COPYDATA. Denna metod kräver samarbete mellan båda applikationerna. Mottagande applikationen måste känna igen formatet på meddelandet och kunna identifiera avsändaren. Metoden använder sig av delat minne och är därför en snabb metod. Metoden har dock vissa begränsningar så som begränsad storlek på meddelandet och enkelriktad trafik d.v.s. information kan inte skickas tillbaka till avsändaren över samma länk. (Msdn, 2012a)

Aktiv kartläggning (File mapping) är en effektiv metod för delning av data mellan två eller flera applikationer på samma dator. Metoden fungerar genom att flytta ett objekt till virtuellt minne som sedan kan nås av alla applikationer. Objektet är vanligtvis en fil som är fysiskt närvarande på hårdskivan. Metoden är speciellt nyttig med större filer då Input/Output mellan hårdskivan och applikationen försvinner. Synkronisering av data är

största problemet med metoden eftersom många program har samtidig tillgång till data. (Microsoft, 2011a)

Mailslot är en simpel metod för överförning av information. Mailslot är en klient-server modell som tillåter avsändning av data lokalt eller inom ett nätverk. Mailslot tillåter utsändning till många olika servrar vilket gör metoden till en bra metod för att skicka korta meddelanden till alla datorer i ett nätverk. NET SEND är ett exempel på en Mailslot program. (Microsoft, 2011a)

Det finns två olika sorters rör för IPC. Ett anonymt rör är ett icke namngivet rör, med enkelriktad trafik som används för överförning av data mellan en förälderprocess och en barnprocess. Anonyma rör kan bara användas lokalt. Ett namngivet rör används för överförning av data mellan lokala processer och mellan processer på olika datorer. En serverprocess skapar ett namngivet rör som klienten känner igen. Klienten kan sedan öppna röret i sin ända och efter det kan data utbytas genom att utföra läs- och skrivoperationer på röret. WCF använder sig av en modifierad version av namngivna rör som heter NetNamedPipe (Microsoft, 2011a)

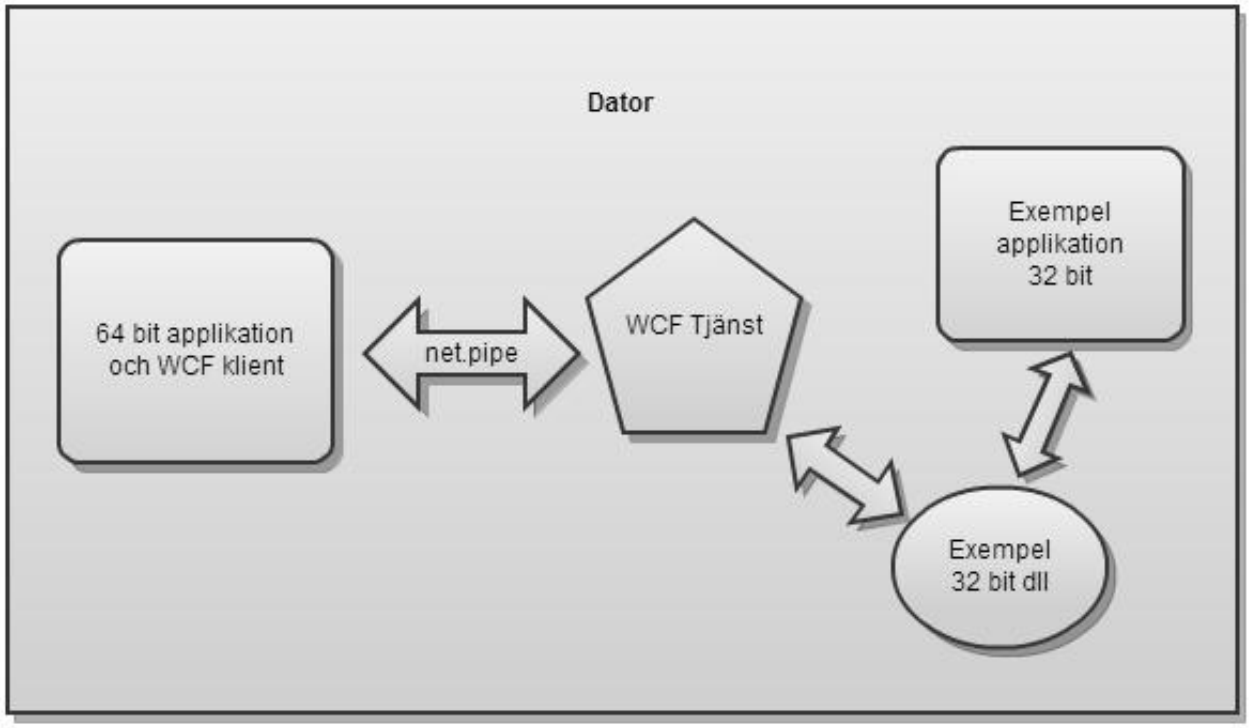
Rör metoden används för överförning av data i examensarbetets exempel i kapitel 5.

Den sista metoden som tas upp är Windows Sockets. Windows Sockets grundar sig på Berkley Sockets (BSD) och kan kommunicera med applikationer på olika operativsystem som även stöder sockets. (Microsoft, 2011a)

5 TILLÄMPNING AV IPC I WCF

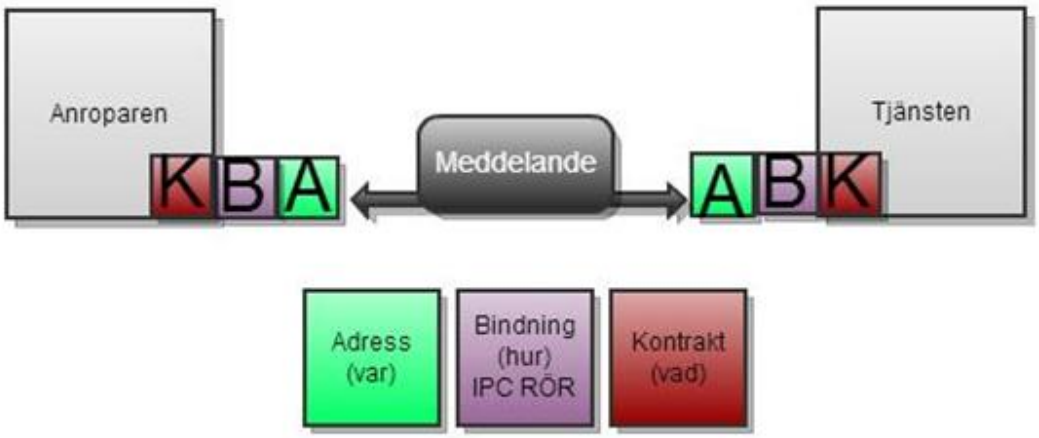
Även om det oftast finns både en 64- och 32-bitars version av moderna program finns det ännu en hel del 32-bitars legacy-applikationer kvar. Problemet med 32-bitars applikationer är att det inte är möjligt att utnyttja en applikation kompilerat med 32-bitar i en 64-bitars applikation utan något måste överföra data mellan applikationerna. Ett sätt att lösa detta problem är WCF och NetNamedPipe (net.pipe) IPC bindningen.

I detta kapitel behandlas hur man skapar en WCF tjänst och en klient. Den skapade WCF tjänsten är en IPC tjänst som sköter om kommunikationen mellan ett 32-bitars DLL-bibliotek och en 64-bitars applikation. Som värd för tjänsten fungerar en konsolapplikation. DLL filen fungerar som ett gränssnitt för en 32-bitars användarapplikation. I figur 10 visualiseras exempelsystemet.



Figur 10 Diagram över tjänst/klient tillämpninge

I figur 11 visas noggrannare kommunikationen mellan WCF tjänsten och WCF klienten.



Figur 11 Kommunikationen i WCF

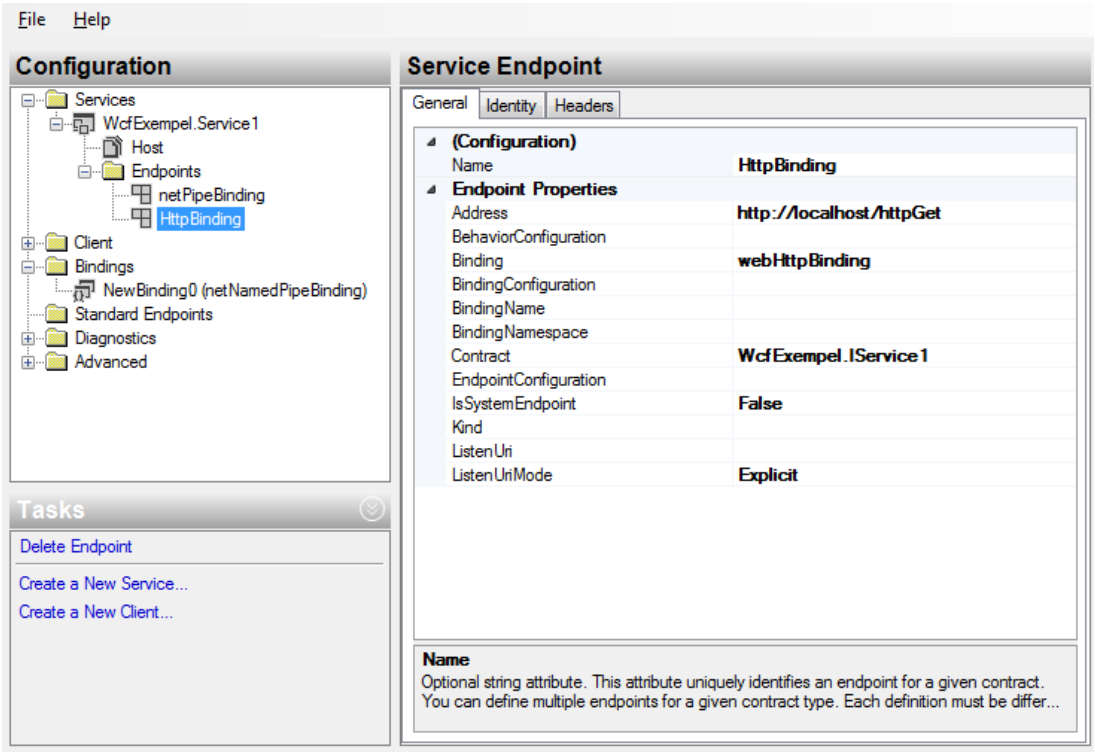
5.1 WCF verktyg

I WCF ramverket ingår olika verktyg som underlättar utvecklingen av tjänster.

5.1.1 SvcConfigEditor

Ett verktyg är SvcConfigEditor.exe. Editorn kan startas inne i Visual studio genom en konfigurationsfil med filändelsen .config. Nyttan av editorn är diskutabel eftersom det ändå krävs full förståelse av alla olika konfigurationsdelar i WCF. (Löwy, 2010)Om utvecklaren väljer att inte använda sig av editorn går det att editera direkt i konfigurations filen. (Löwy, 2010)

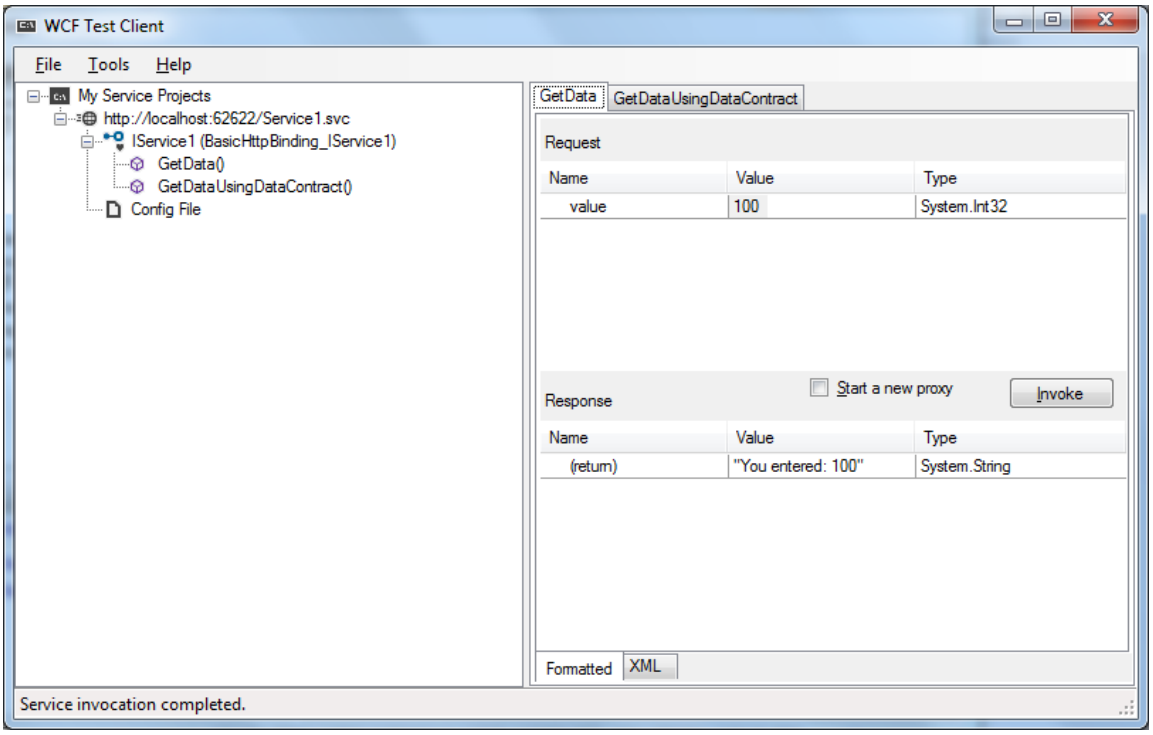
Figur 12 visar hur slutpunktskonfigurationen ser ut i editorn.



Figur 12 SvcConfigEditor används vid editering av konfigurations filer

5.1.2 WcfTestClient

Visual Studio 2012 har inbyggt en WCF test klient som heter WcfTestClient.exe. Test klienten kan hantera olika fundamentala datatyper, klasser och datastrukturer. Klienten kan göra anrop till HTTP, TCP och IPC tjänsteslutpunkter. För att kunna använda test klienten krävs det att tjänsten publicerar metadata. I figur 13 visas hur GetData operationen anropas av klienten och tjänsten svarar med sin respons. (Löwy, 2010)



Figur 13 Användning av WcfTestClient

5.2 Server

Visual Studio 2012 innehåller färdiga mallar för WCF tjänster som kan användas då IIS, WAS eller en Windows tjänst fungerar som värd. I detta exempel används ett anpassat program, en konsolapplikation som grund för exemplet. Konsolapplikationer används oftast för testning och i utvecklingsskede. I exemplet fungerar en konsolapplikation bra p.g.a. att servern bara behövs då användarapplikationen är igång och en konsol kan startas som gömd.

Nästa steg i exemplet var att lägga till en tjänst till konsol applikationen. I Visual Studio 2012 finns ett färdigt objekt som heter WCF Service. Efter att detta objekt tillsättes till Visual Studio lösningen genererades alla filer som krävs för en tjänst att fungera. Filer som krävs för en WCF tjänst är ett gränssnitt där alla tjänstens kontrakt och operationer definieras, samt en tjänsteklass som implementerar gränssnittet och som innehåller all funktionalitet i tjänsten. Sedan generas en XML .config fil dit tjänstens konfiguration kan definieras.

5.2.1 Konfiguration av tjänstens slutpunkter

För att få ett mer beskrivande exempel har tjänsten förutom IPC slutpunkten en TCP slutpunkt som möjliggör att klienter i samma nätverk kan anropa tjänsten. IPC bindningen är programmerad i C# medan TCP bindningen är konfigurerad i XML.

Det som kärvs för att programmera en slutpunkt är att initiera ett ServiceHost objekt och sedan tillsätta en slutpunk för objektet. Kodexemplet visar hur detta görs.

```
//Initierar en ny service host
ServiceHost host = new ServiceHost(typeof(ExempelTjanst));

//Tillsätter en slutpunkt till ServiceHost objektet med kontrakt
gränssnittet, typ av bindning och adressen
host.AddServiceEndpoint(typeof(IExempelTjanst),
                        new NetNamedPipeBinding(),
                        "net.pipe://localhost/pipe");
//Tillsätter en slutpunkt för metadata utbyte för net.pipe
host.AddServiceEndpoint(typeof(ServiceMetadataBegaviour.MexContractName),
                    new MetadataExchangeBIndings.CreateMexNamedPipeBinding(),
                        "net.pipe://localhost/mex");

//Tjänsten börjar lyssna på slutpunkten genom att kalla på Open()
host.Open();
```

Det är möjligt att konfigurera slutpunkter för en tjänst både med kod och XML. Vid konfliktfall åsidosätter programmerade slutpunkter konfigurerade slutpunkter. Konfigurering av TCP slutpunkten i exemplet ser ut på följande sätt:

```
<service name="WCFHost.ExempelTjanst">
        <!-- TCP slutpunkten för tjänsten-->
        <enpoint address="WCFHost/ExempelTjanst"
                 binding="netTcpBinding" contract="WCFHost.IExempelTjanst">
                    <identity>
                        <dns value="localhost" />
                    </identity>
        </endpoint>
        <!-- Metadata slutpunkten för TCP -->
        <enpoint address="mex"
        binding="mexTcpBinding" contract="IMetadataExchange">
        </endpoint>
        <host>
                <!-- Bas adressen för TCP slutpunkten. -->
                <baseAddresses>
                        <add baseAddress="net.tcp://localhost:8232/" />
                </baseAddresses>
        </host>
</service>
```

5.2.2 Kontrakt

Kontrakten för en tjänst definieras oftast i ett separat gränssnitt som sedan implementeras av en tjänsteklass. I detta exempel är det DLL filen som implementeras på nytt i tjänsten för att flytta över DLL gränssnittets funktioner vidare till en 64-bitars applikation. Gränssnittet innehåller både metoder för att ge kommandon åt 32-bitars applikationen och händelser(eng. events). Händelser betyder att då det sker något i användarapplikationen triggas en anmälan till programmet som implementerar DLL gränssnittet, vid detta fall tjänsten. Det är WCF tjänsten som måste skuffa ut meddelandet åt en WCF klienten, detta implementeras i exemplet med Callback operationen.

I koden nedan visas hur Callback gränssnittet programmeras. Då eventet OnNewData avfyras i tjänsten skickas NewData datatypen till alla klienter som är bundna till tjänsten.

```
public interface ICallBack
{
        [OperationContract(IsOneWay = true)]
        void OnNewData(NewData dat);
}
```

Data kontraktet eller datatypen NewData är inte en inbyggd data typ utan är själva definierat som data kontrakt. Data kontraktet NewData är definierat på följande sätt:

```
[DataContract]
public class NewData
{
        private string data; private int id;

        [DataMember]
        public string psData
        {
            get { return status; } set { status = value; }
        }

        [DataMember]
        public int piID
        {
            get { return id; } set { id = value; }
        }
```

```
}
```

En tjänst kräver alltid ett servicekontrakt (eng. ServiceContract) som i sin tur innehåller funktionskontrakt (eng. OperationContract) som kan anropas av klienten. I koden nedan definieras ett service kontrakt med callback gränssnittet ovan. Callback kräver *session mode* för att fungera.

```
[ServiceContract(SessionMode=SessionMode.Required,
CallbackContract = typeof(ICallBack))]
public interface IExempelTjanst
{
    [OperationContract]
    string GetStatus();

    [OperationContract]
    void Initialize();

}
```

5.2.3 Tjänsten

Sista delen för att bygga en tjänst är att implementera själva funktionaliteten av tjänsten. I exemplet görs detta i tjänsteklassen som implementerar IExempelTjanst gränssnittet. Kodexemplet nedan visar tjänstens funktionalitet

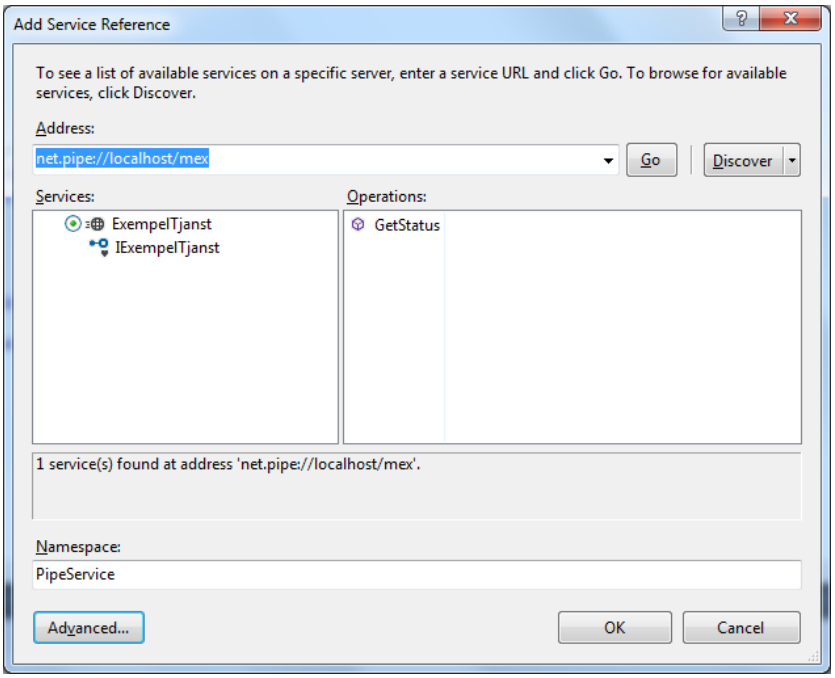
```
public class ExempelTjanst : IExempelTjanst
{
DLL dll;
ICallBack callback;
private NewData dat;

void Initialize()
{
dll = new DLL();
//Initiera callback kanalen med ICallBack gränssnittet
callback = OperationContext.Current.GetCallbackChannel<ICallBack>();
dll.OnNewData += dll_OnNewData;
}

public string GetStatus()
{
string status;
dll.GetStatus(out status);
//Då GetStatus() anropas från en klient returneras detta till klienten
return status;
}


//Då OnNewData eventet triggas körs denna metod som skickar NewData till alla
klienter som är kopplade till tjänsten
public void dll_OnNewData(string data,int id)
{
dat d = new dat();
d.psData = data;
d.piID = id;
callback.OnNewData(d);
}
}
```

5.3 Klient

För att en klient skall kunna anropa en WCF tjänst måste klienten först importera tjänstens kontrakt. Importerande av tjänstens kontrakt sker via metadata utbyte. Ifall det handlar om en klient som är byggd med hjälp av WCF genereras automatiskt en proxy som representerar servicekontraktet. Alla applikationer byggda med .NET kan ju fungera som en WCF klient.

5.3.1 Generering av klient

Lättaste sättet att generera en klientproxy är att i Visual Studio 2012 välja Add Service Referance och med hjälp av tjänstens MEX slutpunkt importera tjänstens kontrakt till klienten. Figur 14 visar hur man lägger till en tjänstereferens.



Figur 14 Generering av klient proxy

5.3.2 Konfigurering och användning av klienten

Om tjänsten inte skulle ha callbacks krävs det bara att initiera klienten och sedan är det möjlig att anropa tjänstens operationer. Då callbacks används krävs det att callback

gränssnittet även implementeras på klientsidan. Kodexemplet nedan visar klient kod för en tjänst som utnyttjar callbacks.

```
public class CallBack : PipeService.IExempelTjanstCallback
{
public void OnNewData(NewData dat)
{
//Här implementeras funktionen då det från tjänsten skickas ett meddelande
till klienten.
}
}

CallBack callback = new CallBack();
InstanceContext context = new InstanceContext();
PipeService.ExempelTjanstClient client = new PipeServ-
ice.ExempelTjanstClient(context);
```

Nu kan klienten anropa tjänstens operationer GetStatus och Initiate med hjälp av client objektet.

6 SLUTSATSER

WCF är ett av de mest anmärkningsvärda sätten att implementera en klient-server modell i .NET och Windows miljöer. WCF möjliggör effektiv konfigurering av tjänster och enkel automatisering för att generera klientkod. Visual Studio innehåller många användbara komponenter som underlättar utvecklingen av WCF tjänster.

Slutsatsen i mitt examensarbete är att WCF utmärkt kan användas för att lösa problemet med att utnyttja 32-bitars programbibliotek i 64-bitars applikationer i Windows miljö. Tyvärr kunde jag inte i examensarbetet bifoga exempel som används i verkliga projekt. Ett sådant skulle tydligare ha visat användningen av WCF/IPC mellan applikationer utvecklade för olika ordlängd. Nu fick de kodsnuttar som ingår i kapitel 5 i stället utgöra den grund man behöver för att effektivt designa, utveckla och konfigurera tjänster med WCF.

Jag skulle gärna ännu mera fördjupat mig i hur REST-tjänster implementeras med WCF och vilka slags lösningar de kan erbjuda. WCF innehåller väldigt mycket sådan funktionalitet som inte beskrivs i detta examensarbete och WCF kan användas av de flesta tjänstebaserade lösningar.

KÄLLOR

Josuttis, Nicolai. 2007, SOA in practice, O'Reilly Media

Kubiatowicz, John. 1998, Integrated Shared-Memory and Message-Passing Communication in the Alewife Multiprocessor Tillgänglig: http://www.cs.berkeley.edu/~kubitron/papers/alewife/pdf/kubi-phdthesis.pdf Hämtad 7.3.2013

Kuhns, Fred. 2004, Core Inter-Process Communication Mechanisms [www] Tillgänglig:http://www.cs.wustl.edu/~fredk/Courses/cse422/sp04/Lectures/ipc.ppt Hämtad 7.3.2013

Lublinsky, Boris. 2007, Defining SOA as an architectural style [www] Tillgänglig: http://www.ibm.com/developerworks/architecture/library/ar-soastyle/ Hämtad 27.2.2013

Löwy, Juval. 2010, Programming WCF Services, Third Edition, O'Reilly Media

Microsoft, 2011a. Interprocess Communications (Windows) [www] Tillgänglig: http://msdn.microsoft.com/en-us/library/windows/desktop/aa365574(v=vs.85).aspx Hämtad 6.3.2013

Microsoft, 2012a. Windows Communication Foundation [www] Tillgänglig: http://msdn.microsoft.com/en-us/library/dd456779.aspx Hämtad 2.4.2013

Microsoft, 2012b. .NET Framework 4.5 [www] Tillgänglig: http://msdn.microsoft.com/en-us/library/vstudio/w0x726c2.aspx Hämtad 24.3.2013

Msdn, 2012a. Using WM_COPYDATA to marshal message parameters since the window manager otherwise doesn't know how [www] Tillgänglig: http://blogs.msdn.com/b/oldnewthing/archive/2012/10/19/10360980.aspx

Oasis Reference Model for Service Oriented Architectures, Committee Draft 1.0, 7.2.2006 Tillgänglig: http://www.oasisopen.org/committees/download.php/16587/wd-soa-rmcd1ED.pdf Hämtad 27.2.2013

Rosen,Mile; Lublinsky, Boris; Smith, Kevin; Balcer, Mark. 2008, Applied SOA - serivce-oriented architecture and design strategies, Wiley Publishing

Wikipedia, 2012a. Inter-process communication [www] Tillgänglig: http://en.wikipedia.org/wiki/Inter-process_communication Hämtad 6.3.2013

Wikipedia, 2013a. .NET Framework [www] Tillgänglig: http://en.wikipedia.org/wiki/.NET_Framework Hämtad 24.3.2013