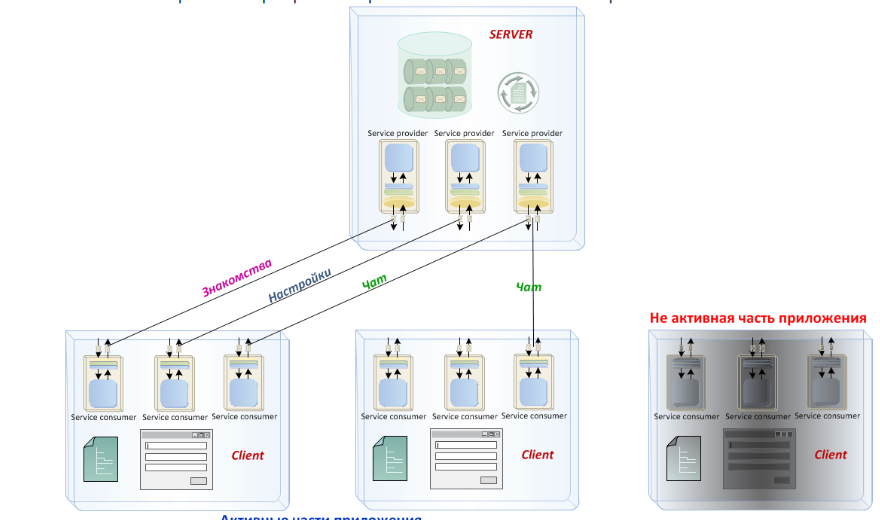
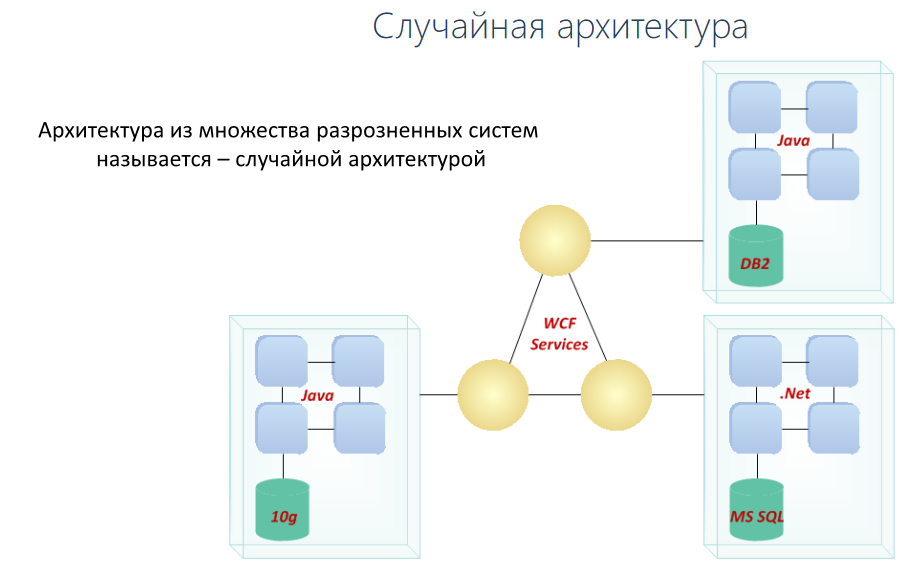
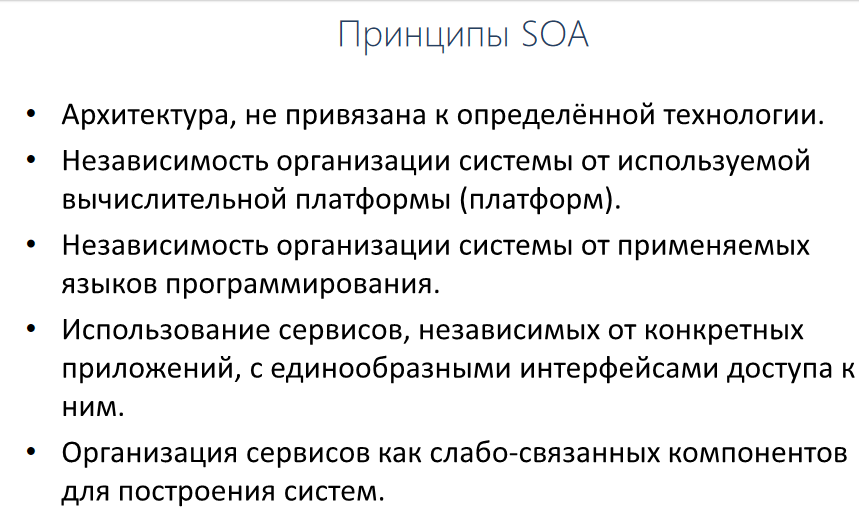
***WCF*** – это технология, предназначенная для проектирования, построения, сопровождения и модификации распределенных приложений.



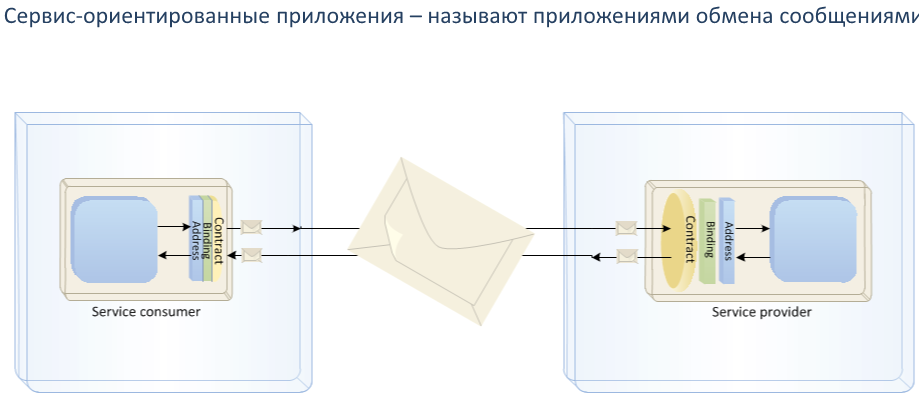


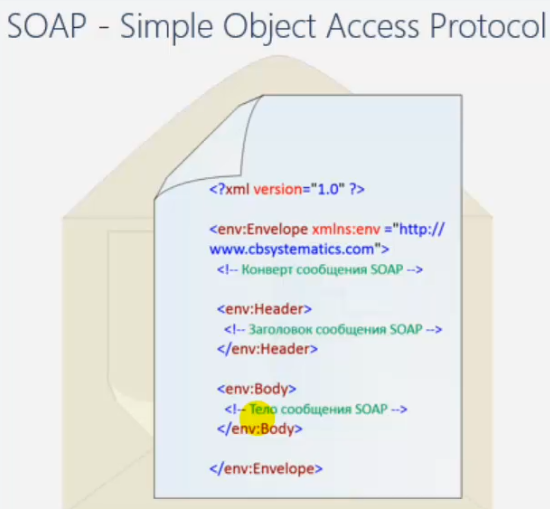




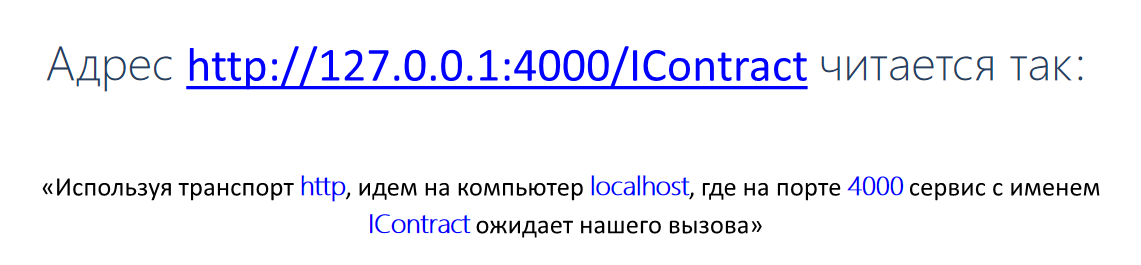


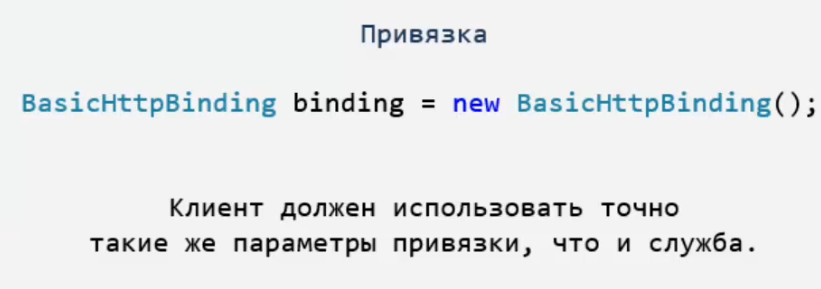




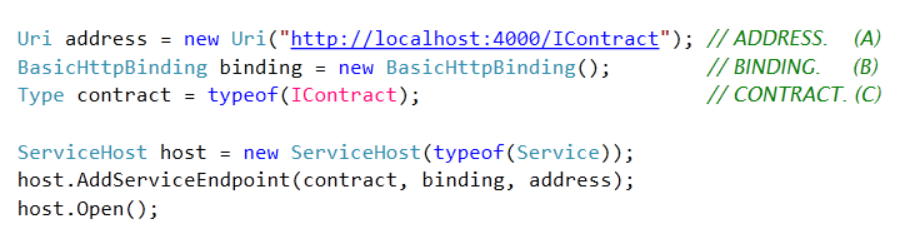


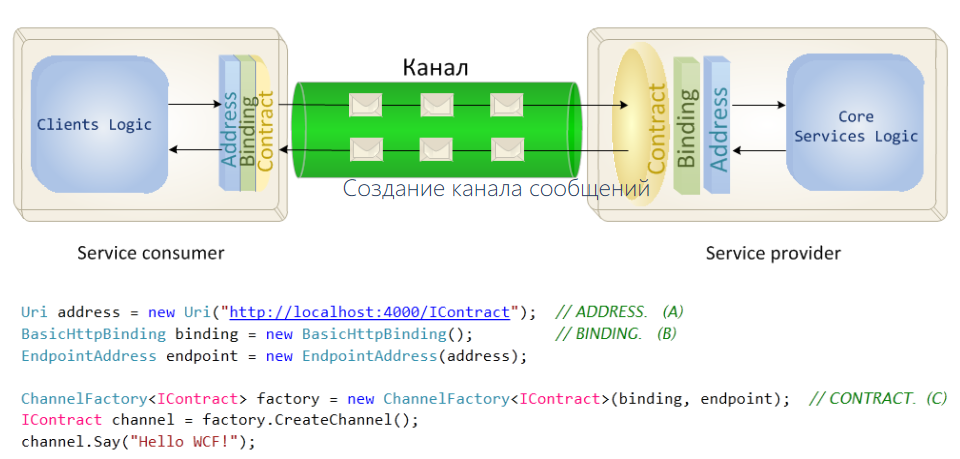


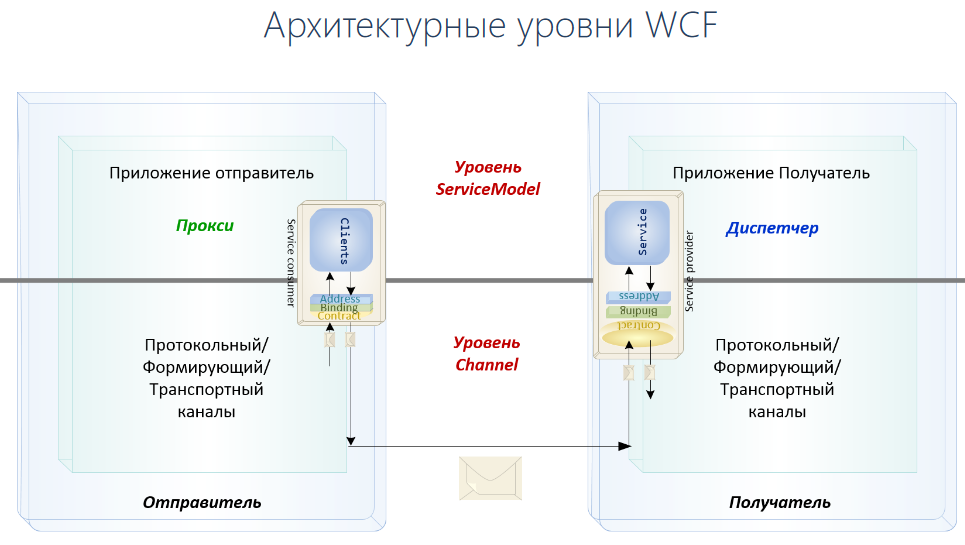












\*\*\*

# Клиент

namespace Client

{

[ServiceContract]

interface IContract

{

[OperationContract]

void Say(string name, string input);

}

}

class Client

{

static void Main(string[] args)

{

Console.Title = "CLIENT";

Console.WriteLine("Введите имя пользоветеля");

string user = Console.ReadLine();

// Указание, где ожидать входящие сообщения.

Uri address = new Uri("http://localhost:4000/IContract"); // ADDRESS. (A)

// Указание, как обмениваться сообщениями.

BasicHttpBinding binding = new BasicHttpBinding(); // BINDING. (B)

// Создание Конечной Точки.

EndpointAddress endpoint = new EndpointAddress(address);

// Создание фабрики каналов.

ChannelFactory<IContract> factory = new ChannelFactory<IContract>(binding, endpoint); // CONTRACT. (C)

// Использование factory для создания канала (прокси).

IContract channel = factory.CreateChannel();

// Использование канала для отправки сообщения получателю.

while (true)

{

Console.WriteLine("Введите сообщение");

string message = Console.ReadLine();

channel.Say(user,message);

}

// Задержка.

Console.ReadKey();

}

}

# Сервер

[ServiceContract]

interface IContract

{

[OperationContract]

void Say(string name, string input);

}

class Service : IContract

{

// Реализация контракта - IContract.

public void Say(string name,string input)

{

Console.WriteLine("{0}: {1}", name,input);

}

}

class Server

{

static void Main()

{

Console.Title = "SERVER";

// Указание адреса, где ожидать входящие сообщения.

Uri address = new Uri("http://localhost:4000/IContract"); // ADDRESS. (A)

// Указание привязки, как обмениваться сообщениями.

BasicHttpBinding binding = new BasicHttpBinding(); // BINDING. (B)

// Указание контракта.

Type contract = typeof(IContract); // CONTRACT. (C)

// Создание провайдера Хостинга с указанием Сервиса.

ServiceHost host = new ServiceHost(typeof(Service));

// Добавление "Конечной Точки".

host.AddServiceEndpoint(contract, binding, address);

// Начало ожидания прихода сообщений.

host.Open();

Console.WriteLine("Приложение готово к приему сообщений.");

Console.ReadKey();

// Завершение ожидания прихода сообщений.

host.Close();

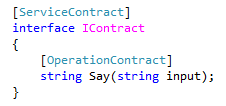
}

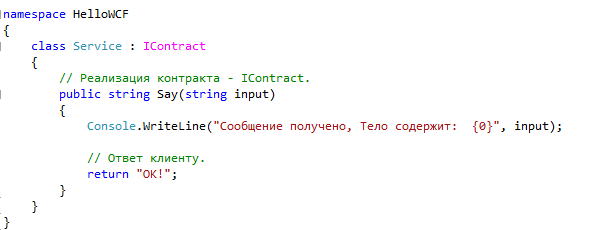
}

\*\*\*

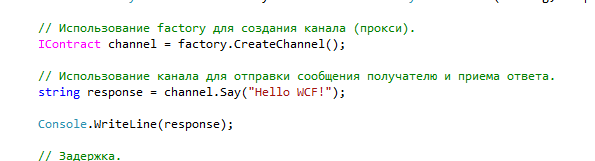
Приложение с возвращаемым значением

Сервер





Клиент



Основная функциональность WCF

разделяется на 10 категорий

1. Независимая модификация
2. Асинхронный однонаправленный обмен сообщениями

**Возможность написания приложений с применением вызовов функций типа «Запрос-Ответ».**

3. Платформенная консолидация

**Объединение возможностей многих предыдущих технологий в единую программную модель.**

4. Безопасность

* •Аутентификация
* •Авторизация
* •Конфиденциальность
* •Целостность
* •Безопасность на уровне транспорта
* •Безопасность на уровне сообщений

5. Надежность

**В распределенных вычислениях, надежная доставка сообщений называется гарантированной.**

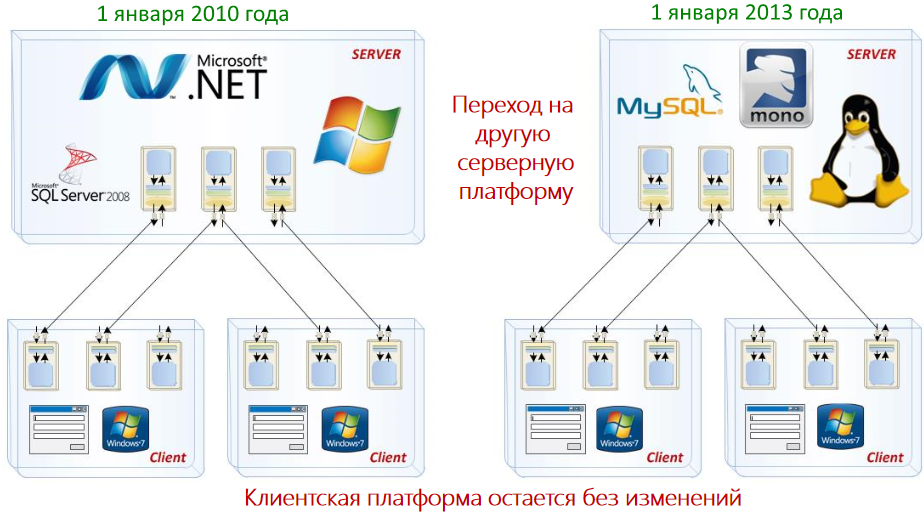
WCF содержит механизмы, обеспечивающие три вида гарантии надежной доставки сообщений:

* 1.Не более одного раза.
* 2.Хотя бы один раз.
* 3.По порядку.

6. Поддержка транзакций

7. Сетевое взаимодействие (Interoperability)

**Способность системы менять программную или аппаратную платформу, не затрагивая других участников сценария распределенных вычислений.**



8. Производительность

**Производительность при взаимодействии между двумя WCF приложениями выше,**

**чем между WCF-приложением и Web-сервисом на базе Java.)**

9. Расширяемость

**WCF - предоставляет возможности**

**для расширения своей архитектуры.**

WCF позволяет создавать свои собственные:

* •Транспорт
* •Каналы
* •Привязки
* •Кодировки
* •Архитектурные парадигмы

10. Возможности конфигурирования

* **WCF - предоставляет возможности конфигурирования посредством**
* **конфигурационных файлов на базе XML.**
* **App.config**
* Возможность конфигурирования позволяет сотрудникам, не являющимся разработчиками, сопровождать приложение и изменять порядок его действий без привлечения команды разработчиков.

Выводы

* ***WCF*** позволяет проектировать, строить, отлаживать и сопровождать распределенные системы быстрее и более функционально, чем это было раньше.
* ***WCF*** объединяет множество технологий: RPC, COM+, Remoting, ASMX, WSE и MSMQ.
* ***WCF*** расширяема.