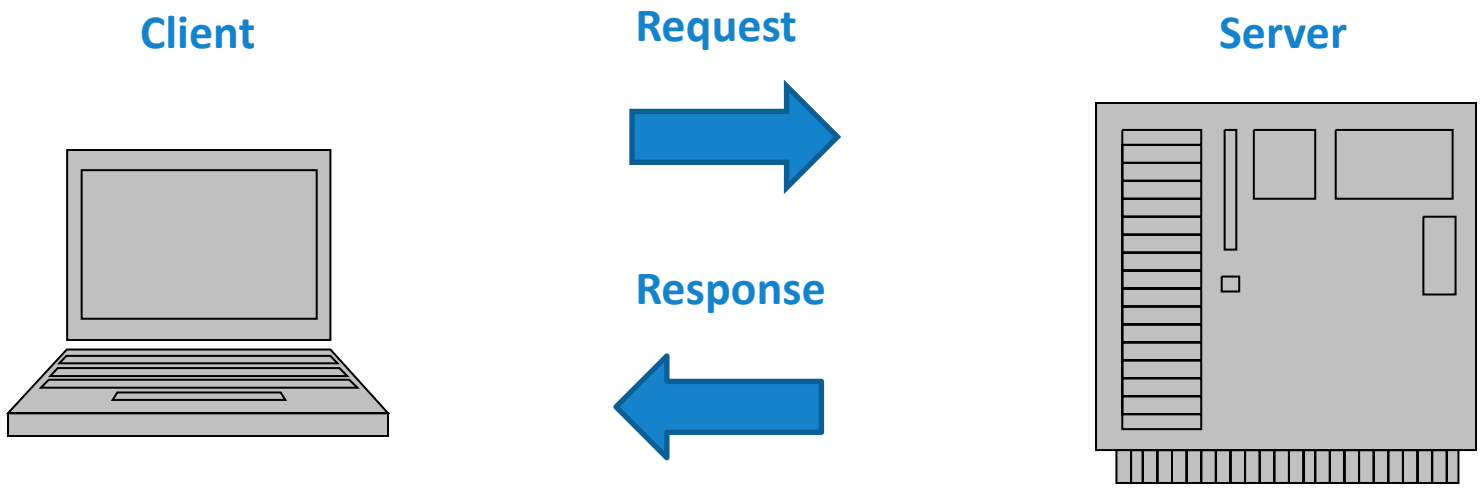
alpha Software®

Pulling In the .NET

Connecting to SOAP and WCF Web Services from XBasic

Kurt Rayner
Vice President Research & Development
Alpha Software

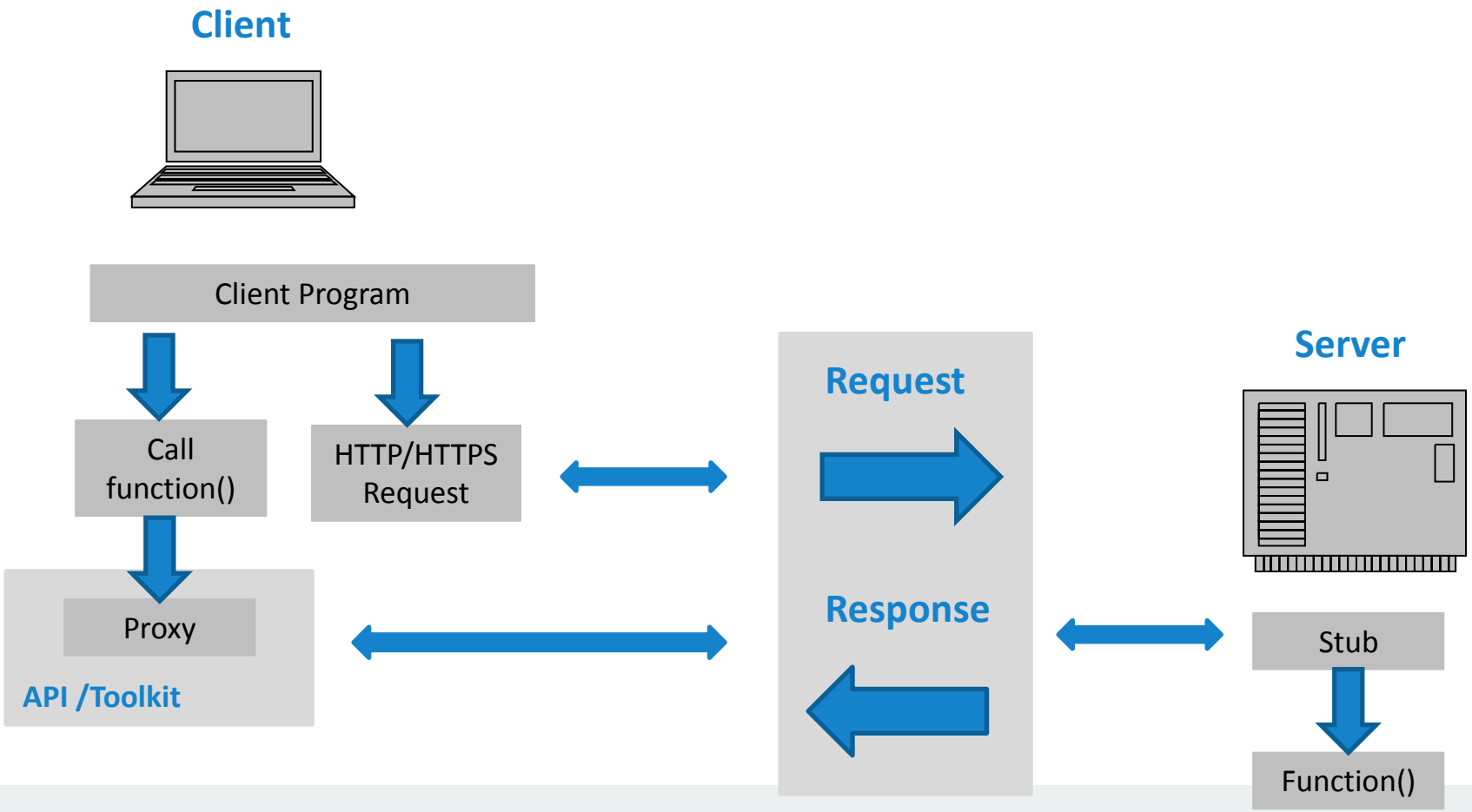
Web Services



Web Services

- A "***standardized***" way of communicating between a *client* and a *server* program ***using web protocols***
- Have a lot of ***history*** - DCE/RPC -> APPC -> CORBA -> DCOM -> CGI -> ***SOAP*** -> ***RESTful*** -> ?
- ***Interoperable*** – *Programming language, run-time, operating system and hardware independent*

Web Services



Web Services

What has to happen:

- Find the service (TCP/IP address and port)
- Serialize data (bit formats and character sets)
- Make a specific request/send response
- De-serialize data (back to objects I know about)

Oh… yeah…, and while you're at it…

- Do it all securely!

Note: APIs make all of this look like a function call

SOAP - Simple Object Access Protocol

- Remote procedures, but you can ***pass objects***
- Callable using ***HTTP/HTTPS*** (ports 80/443)
- Formatted as ***XML*** (Extensible Markup Language)
- ***Self describing*** in a standard format called ***WSDL*** (Web Service Definition Language) – also XML
- Supposed to be discoverable through ***UDDI*** (Universal Description, Discovery and Integration) and Microsoft's DISCO (Web Service Discovery Tool)
- Led to a set of services called **WS-*** and encouraged by W3C
  - WSHTTPBinding in Microsoft WCF

Simple Object Access Protocol - SOAP

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
        <soap:Header> </soap:Header>
        <soap:Body>
                <m:GetStockPrice xmlns:m="http://www.example.org/stock/Surya">
                <m:StockName>IBM</m:StockName>
                </m:GetStockPrice>
        </soap:Body>
</soap:Envelope>
```

Ref: https://en.wikipedia.org/wiki/SOAP

WSDL
Web Service Definition Language

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://www.webserviceX.NET"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.webserviceX.NET" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/">
<wsdl:types>
    <s:schema targetNamespace="http://www.webserviceX.NET" elementFormDefault="qualified">
    <s:element name="GetWeather"><s:complexType><s:sequence>
    <s:element name="CityName" type="s:string" maxOccurs="1" minOccurs="0"/><s:element name="CountryName" type="s:string" maxOccurs="1"
    minOccurs="0"/></s:sequence></s:complexType></s:element>
    <s:element name="GetWeatherResponse"><s:complexType><s:sequence>
    <s:element name="GetWeatherResult" type="s:string" maxOccurs="1" minOccurs="0"/></s:sequence> </s:complexType></s:element>
    <s:element name="GetCitiesByCountry"><s:complexType><s:sequence>
    <s:element name="CountryName" type="s:string" maxOccurs="1" minOccurs="0"/></s:sequence></s:complexType></s:element>
    <s:element name="GetCitiesByCountryResponse"><s:complexType><s:sequence>
    <s:element name="GetCitiesByCountryResult" type="s:string" maxOccurs="1" minOccurs="0"/></s:sequence></s:complexType></s:element>
    <s:element name="string" type="s:string" nillable="true"/>
    </s:schema>
</wsdl:types>
```

… (goes on for pages)…

What about RESTful Services?

REpresentational **S**tate **T**ransfer

- Attempt to move away from procedures to transferring the "***state***" of a "***resource***" (from verbs to nouns).
- Identify the resource as part of the ***URI*** (uniform resource identifier)
- Revives the old ***CRUD*** (Create, Read, Update, Delete) acronym and maps it (roughly) to HTTP verbs POST, GET, PUT, DELETE)
  - Note: PUT/POST have overlapping use.
    - POST can be used to create an object or add a member to a collection
    - PUT updates a resource completely (idempotent) – "create if it doesn't exist"
  - Use OPTIONS HTTP verb to find out the verbs that are supported (Allow header)
- Interaction is stateless
- ***XML*** or ***JSON***

Ref: http://stackoverflow.com/questions/630453/put-vs-post-in-rest

What about RESTful Services?

POST /articles HTTP/1.1

<article>
<title>Blue Stapler</title>
<price currency="eur">7.50</price>
</article>

Ref: http://restcookbook.com

What about RESTful Services?

```
POST /articles HTTP/1.1

{
title: “Blue Stapler”,
price:      {
            currency: “eur”,
            value: 7.50
            }
}
```

What about RESTful Services?

```
var my_JSON_object;
var http_request = new XMLHttpRequest();
http_request.open("GET", url, true);
http_request.onreadystatechange = function ()
            {
            var         done = 4,
                        ok = 200;
            if (        http_request.readyState         === done
               &&       http_request.status             === ok)
                        {
                        my_JSON_object = JSON.parse(http_request.responseText);
                        }
            };
http_request.send(null);
```

Ref: https://en.wikipedia.org/wiki/JSON

Service Type versus Data Format

JSON

```
{
   "ID": "1",
   "Name": "M Vaqqas",
   "Email": fred@gmail.com,
   "Country": "India“
}
```

XML

```
<Person>
<ID>1</ID>
<Name>M Vaqqas</Name>
<Email>fred@gmail.com</Email>
<Country>India</Country>
</Person>
```

Ref: http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

Windows Communication Foundation (WCF)

- Proprietary Microsoft technology
  - Open sourced this year – https://www.dotnetfoundation.org/blob/wcf-is-open-source
- Intended to create a highly configurable architecture
- Supports both SOAP and RESTFul APIs with XML and JSON data
- **ABC**
  - Address – Where is the service?
  - Binding – How are we getting there?
  - Contract – What is it we expect to be doing?
- Client - best to use a ***proxy*** assembly (generated using svcutil.exe or by Alpha Anywhere)
- Server – creation is tightly integrated with Visual Studio

Creating and Using Web Service Proxies in XBasic

Steps

- Generate the web service proxy – once
- Add the new assembly to the XBasic type system – once per running application
- DIM the web service client object and use it to make calls to the service – as needed

Generating .NET Web Service Clients

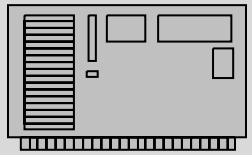
Get WSDL

Retrieve namespaces

Create proxy code

Create .NET assembly

http://myhost.com/myservice.svc?WSDL



```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.webserviceX.NET"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns
```

myproxy.dll

Step 1: Generate the Web Service Proxy

```
dim Sv as DotNet::Services
Sv.GenerateWebServiceClientFromURL(
        "http://hostname.com/service.svc?WSDL",
         "c:\temp\myproxy.dll")
```

Notes:

- GenerateWebServiceClientFromURL returns true or false. Check the CallResult property.
- Do this once. Save the file and install it on your server in the executable path.
- The ".svc" extension is used by Microsoft for its web services. You may also find ".asmx", ".php", ".soap.2", ".cfc", another one, or even none at all.

Step 2: Add the Assembly to the XBasic Type System

```
dim Sv     as DotNet::Services
dim Assy as DotNet::AssemblyReference
Assy.FileName = "c:\temp\myproxy.dll"
Sv.RegisterAssembly("::Serv", Assy)
```

Notes:

- RegisterAssembly returns true or false. Check the CallResult property.
- Do this once for each running web application. Use Type::Definition::Exists("::Serv")
- Use a namespace ("::Serv" is probably not a good namespace!) that is unique on your server. In many cases, the root namespace in the assembly is unique enough and you can use "::" to install it in the root of the Alpha Anywhere type system.

Step 3: DIM the Client

dim Client as *::Serv::Client*

Notes:

- Do this as needed. You do not have to hold on to the Client variable.
- Don't worry that this looks too easy! We'll add more code later…

Using the Web Service Client

Once the client is dimmed, you can call its functions as with any .NET assembly.

It ***could*** be as simple as:

```
dim Client as ::Serv::Client
Result = Client.SumValues(1,2,3,4,5)
```

…but it probably won't…

Many web services use a request/response pattern.

Request/Response Pattern

Each function has an associated request and response class:

- Request object is populated and passed to the function
- Response object is returned

```
DIM Request as MyNamespace::MyFunctionRequest
DIM Response as MyNamespace::MyFunctionResponse
Request.SomeValue = ‘value’
Response = MyClient.MyFunction(Request)
```

Request/Response Pattern

```
dim Request as WorldPay::ProcessCCSaleRequest
Request.ccinfo = new WorldPay::CreditCardInfo()
Request.ccinfo.acctid       = "TEST0"
Request.ccinfo.ccname       = "Tony Test"
Request.ccinfo.ccnum        = "5454545454545454"
Request.ccinfo.amount       = 3.00
Request.ccinfo.expmon       = 03
Request.ccinfo.expyear      = 2010

Response = Client.ProcessCCSale(Request)
?response.ProcessCCSaleReturn
authcode = "DECLINED:1000420001:VALID PIN REQUIRED:"
```

Fine Tuning Behavior

- Security
- Configuration

But First…Bindings and Endpoints

Use an alternate constructor to specify the binding and endpoint (address). Remember ABC?

- **EndPoint** - System::ServiceModel::EndPointAddress

```
DIM e as System::ServiceModel::EndpointAddress = \
        new System::ServiceModel::EndPointAddress("https://mydestination/foo.svc")
```

- **Binding** - System::ServiceModel::<binding type>
  - WSHttpBinding
  - BasicHttpBinding
  - WSDualHttpBinding, WSFederationHttpBinding, NetTcpBinding, NetNamedPipeBinding, NetMsmqBinding, NetPeerTcpBinding

```
DIM b as System::ServiceModel::WSHttpBinding

DIM MyClient as ::Serv::Client = new ::Serv::Client(b, e)
```

Windows Security with WCF

Use the binding object to set security information:

```
dim b as System::ServiceModel::WSHttpBinding
b.Security.Mode                                = System::ServiceModel::SecurityMode::None
b.Security.transport.ClientCredentialType = System::ServiceModel::HttpClientCredentialType::Windows
b.Security.Transport.ProxyCredentialType = System::ServiceModel::HttpProxyCredentialType::None
b.Security.Transport.Realm                     = ""
b.Security.Message.ClientCredentialType  = System::ServiceModel::HttpClientCredentialType::Windows
b.Security.Message.NegotiateServiceCredential = .t.

dim e as System::ServiceModel::EndPointAddress = \
            new System::ServiceModel::EndPointAddress("http://…")

dim MyClient as ::Serv::MyClient = new ::Serv::MyClient(b, e)
```

Basic Security with WCF

Use the binding object to set security information:

```
dim b as System::ServiceModel::BasicHttpBinding
b.MessageEncoding                          = System::ServiceModel::WSMessageEncoding::Text
b.Security.Mode                            = System::ServiceModel::SecurityMode::Transport
b.Security.Transport.ClientCredentialType  = System::ServiceModel::HttpClientCredentialType::None
b.Security.Transport.ProxyCredentialType   = System::ServiceModel::HttpProxyCredentialType::None

dim e as system::ServiceModel::EndPointAddress = \
        new System::ServiceModel::EndPointAddress("http://xxxxxxxxx")

dim MyClient as ::Serv::MyClient = new ::Serv::MyClient(b, e)
```

Changing Timeouts

Use the binding object to set timeouts:

```
dim b as System::ServiceModel::BasicHttpBinding
b.OpenTimeout    = new System::TimeSpan(0,0,10)     ' Connect
b.ReceiveTimeout = new System::TimeSpan(0,0,20)     ' Inactivity
b.SendTimeout    = new System::TimeSpan(0,1,0)      ' Write Operation
b.CloseTimeout   = new System::TimeSpan(0,0,15)     ' Close Operation

dim e as system::ServiceModel::EndPointAddress = \
        new System::ServiceModel::EndPointAddress("http://xxxxxxxxx")

dim ScriptService as ::Serv::MyClient = new ::Serv::MyClient(b, e)
```

Changing Quotas and Limits

Use the binding object to set timeouts:

```
dim b as System::ServiceModel::BasicHttpBinding
b.MaxReceivedMessageSize                = 10485760
b.maxBufferSize                         = 10485760
b.MaxBufferPoolSize                     = 10485760
b.ReaderQuotas.maxDepth                 = 32
b.ReaderQuotas.MaxStringContentLength   = 10485760
b.ReaderQuotas.MaxArrayLength           = 16384
b.ReaderQuotas.MaxBytesPerRead          = 4096
b.ReaderQuotas.maxNameTableCharCount    = 16384

dim e as system::ServiceModel::EndPointAddress = \
        new System::ServiceModel::EndPointAddress("http://xxxxxxxxx")

dim ScriptService as ::Serv::MyClient = new ::Serv::MyClient(b, e)
```

But wait…there's more…!

After you DIM the client you may need to set the maximum number of items in the object graph for each operation:

```
Operations = MyClient.Endpoint.Contract.Operations

for i = 1 to Operations.Count
        Operation = Operations[i]
        Behaviors = Operation.Behaviors

        Enumerator = Behaviors.GetEnumerator()
        while Enumerator.MoveNext()
                Current = Enumerator.Current
                if Current.gettype().FullName = \
                        "System.ServiceModel.Description.DataContractSerializerOperationBehavior"
                    Current.MaxItemsInObjectGraph = 2147483647
                end if
        end while
next
```

Getting More Help

- The web service provider should document examples, but will rarely show how to use XBasic directly. The C# samples tend to be the easiest to convert.
  - Watch for subtle differences like using "**::**" for XBasic namespaces rather than "**.**" in C#. Instance references are the same as XBasic.
- Use the interactive window to explore the type space of the new assembly along with the documentation from by the web service provider.
- Browse to the base address of the web service to see if it has a help page.
  http://www.webservicex.com/currencyconvertor.asmx

Questions?