Providing RPG Web Services on IBM i

Presented by

Scott Klement

http://www.scottklement.com

"A computer once beat me at chess, but it was no match for me at kick boxing." — Emo Philips

Our Agenda

Agenda for this session:

1. Introduction
   - What's a web service?
   - Why web services?
   - Types (REST/SOAP/XML/JSON)
2. SOAP web service with IBM's IWS
3. REST web service with IBM's IWS
4. Writing your own from the ground-up with Apache.
5. Discussion/wrap-up

I am a Web Service. What Am I?

A routine that can be called over a TCP/IP network.

- A callable routine. (Program? Subprocedure?)
- Callable over a TCP/IP Network. (LAN? Intranet? Internet?)
  ….can also be called from the same computer.
- Using the HTTP (or HTTPS) network protocol

Despite the name, not necessarily "web"

- different from a "web site" or "web application"
- input and output are via "parameters" (of sorts) and are for programs to use. No user interface -- not even a browser.
- can be used *from* a web application (just as an API or program could) either from JavaScript in the browser, or from a server-side programming language like RPG, PHP, .NET or Java
- but is just as likely to be called from other environments… even 5250!

Write Once, Call From Anywhere

In other words…   Services Oriented Architecture (SOA).

- Your business logic (business rules) are implemented as a set of "services" to any caller that needs them.
- Web services are only <u>*one of many*</u> ways to implement SOA. Don't believe the hype!

Callable from anywhere

- Any other program, written in (just about) any language.
- From the same computer, or from another one.
- From the same office (data center), or from another one.
- From folks in the same company, or (if desired) any of your business partners.   Even the public, if you want!

RPG can function as either a *provider* (server) or a *consumer* (client)

…this session focuses on providing.

What is a Web Service?

A “program call” (or subprocedure call) that works over the Web.

- Very similar in concept to the CALL command.

```
CALL PGM(EXCHRATE) PARM('us' 'euro' &DOLLARS &EUROS)
```

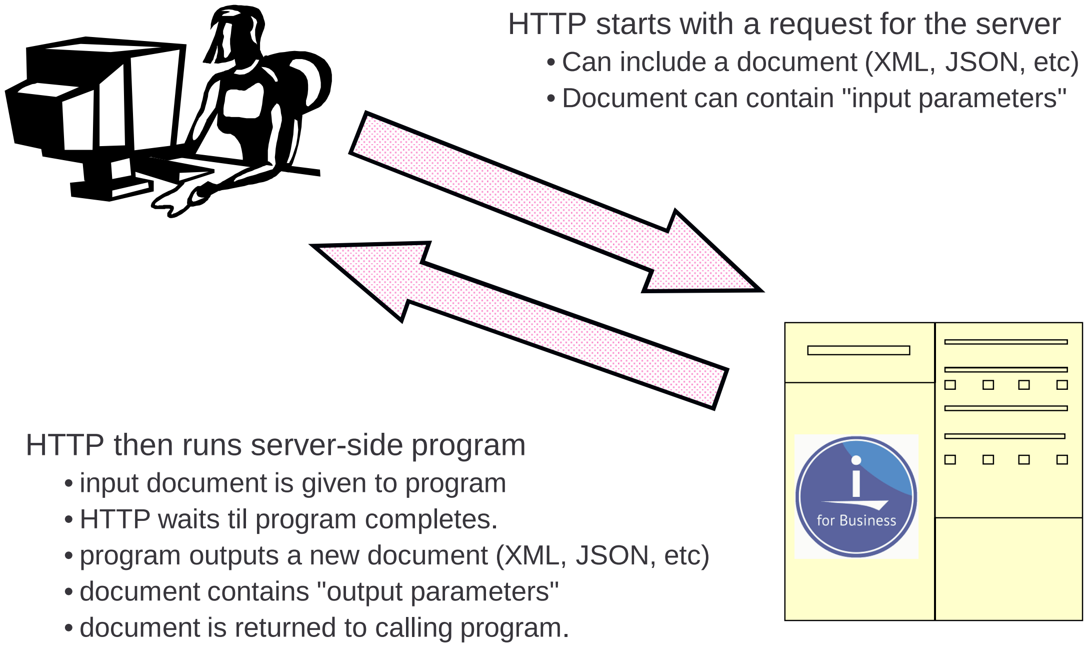
- Runs over the Web, so can be called from programs on other computers anywhere in the world.
- Maybe a web front-end?
  - (Java, .NET, PHP, JavaScript framework, even another RPG.)
- Maybe a thick-client program (windows program, mobile app, etc.)

To be called from a program

Designed to be called from other programs, instead of interfacing directly with the user.

- Web services do not display a screen, or prompt a user
- All input comes from “parameter” data.
- All output is sent via “parameter” data
- Often referred to as an “API”

How Do They Work?



HTTP starts with a request for the server
- Can include a document (XML, JSON, etc)
- Document can contain "input parameters"

HTTP then runs server-side program
- input document is given to program
- HTTP waits til program completes.
- program outputs a new document (XML, JSON, etc)
- document contains "output parameters"
- document is returned to calling program.

REST vs SOAP

SOAP: "Simple Object Access Protocol"
The "old" way. Not as common anymore, but has some advantages.
- URL identifies the web services server
- Input/output documents are always XML in SOAP format
- The “verb” (or action to perform) is given in a separate “soap-action” keyword.
- An accompanying WSDL document describes the SOAP details, including networking details and schema
- Much more complex than REST, but…
- Many more tools are available (vs REST) which can make SOAP easier to code than REST.

REST: "REpresentative State Transfer"
The "new" way. Most new web services use this method.
- URL identifies a “resource” to work with.
- Input/output documents may be in any format. (Most commonly XML or JSON)
- Often, all input is within the URL
- Technically, the HTTP method should be the “verb” (type of action to take), but many web services do not use this approach, and still refer to themselves as REST
- Much simpler/runs faster than SOAP.

XML vs. JSON

Both XML and JSON are widely used in web services:

- Self-describing
- Can make changes without breaking compatibility
- Available for all popular languages / systems

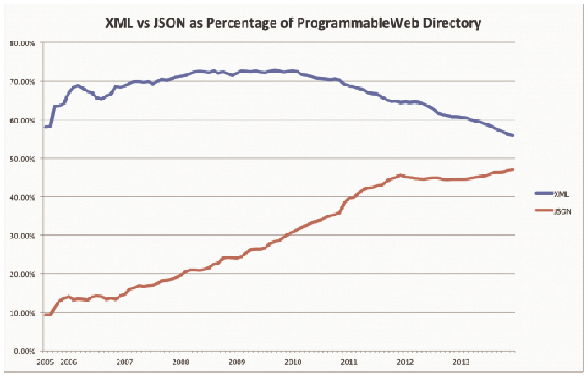
XML:

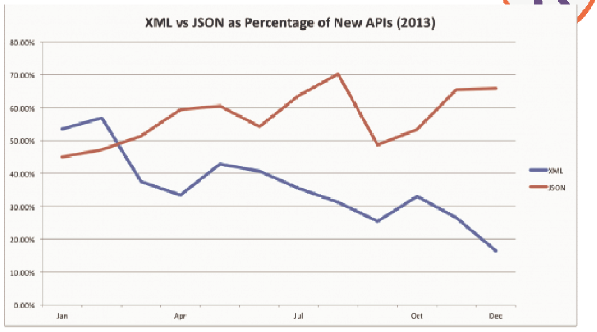
- Has schemas, namespaces, transformations, etc.
- Has been around longer.
- Only format supported in SOAP

JSON:

- Natively supported by all web browsers
- Results in smaller documents (means faster network transfers)
- Parses faster.

JSON is "Taking Over"





In a 2013 study done by the ProgrammableWeb (web service directory and community), we can see JSON growing while XML is declining.

As a percentage of the overall directory (left) XML is higher, but it's close.

For new APIs, JSON is much higher

JSON and XML to Represent a DS

```
D list            ds                  qualified
D                                     dim(2)
D   custno                       4p 0
D   name                        25a
```

Array of data structures in RPG…

```
[
  {
    "custno": 1000,
    "name": "ACME, Inc"
  },
  {
    "custno": 2000,
    "name": "Industrial Supply Limited"
  }
]
```

Array of data structures in JSON

```
<list>
  <cust>
    <custno>1000</custno>
    <name>Acme, Inc</name>
  </cust>
  <cust>
    <custno>2000</custno>
    <name>Industrial Supply Limited</name>
  </cust>
</list>
```

Array of data structures in XML

Without Adding Spacing for Humans

```
[{"custno": 1000,"name": "ACME, Inc"},{"custno": 2000,
"name": "Industrial Supply Limited"}]
```

92 bytes

```
<list><cust><custno>1000</custno><name>ACME, Inc</name
></cust><cust><custno>2000</custno><name>Industrial S
upply Limited</name></cust></list>
```

142 bytes

In this simple "textbook" example, that's a 35% size reduction.

50 bytes doesn't matter, but sometimes these documents can be megabytes long – so a 35% reduction can be important.

…and programs process JSON faster, too!

IBM's Integrated Web Services Server

IBM provides a Web Services tool with IBM i at no extra charge!

The tool takes care of all of the HTTP and XML work for you!

It's called the *Integrated Web Services* tool.

http://www.ibm.com/systems/i/software/iws/

- Can be used to provide web services
- Can also be used to consume them -- but requires in-depth knowledge of C and pointers -- I won't cover IBM's consumer tool today.

Requirements:

- IBM i operating system, version 5.4 or newer.
- 57xx-SS1, opt 30: QShell
- 57xx-SS1, opt 33: PASE
- 57xx-JV1, opt 8: J2SE 5.0 32-bit (Java)
- 57xx-DG1 -- the HTTP server (powered by Apache)

Make sure you have the latest cum & HTTP Sever group PTFs installed.

Let's Get Started!

The HTTP server administration tool runs in a special HTTP server called *ADMIN, and you use it from your browser.

- If this isn’t already started, you can start it with:

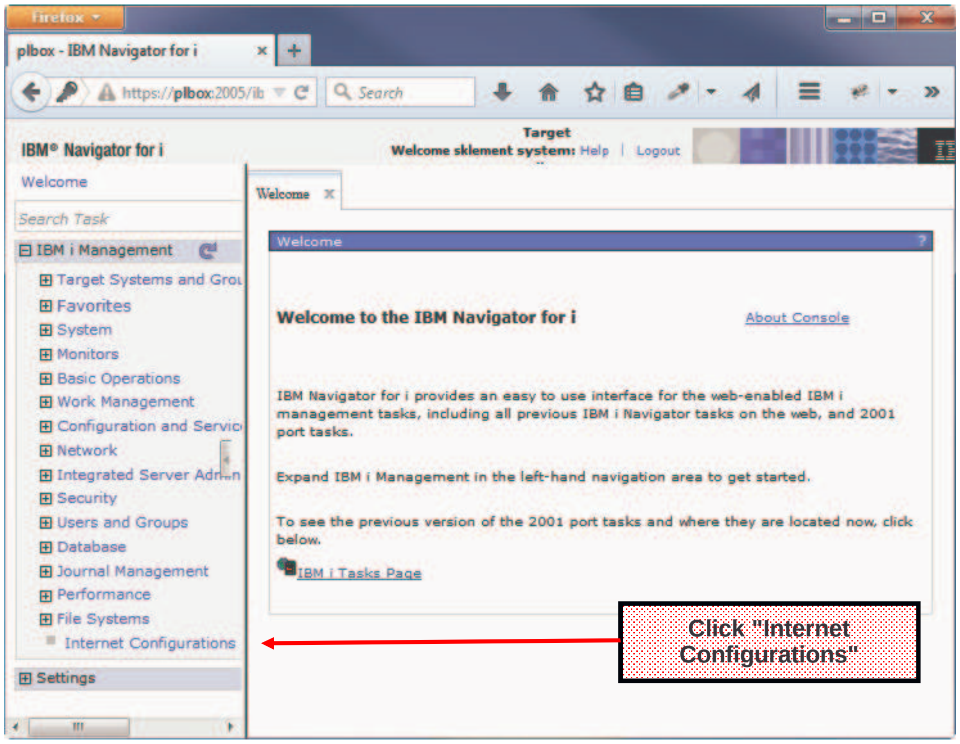
```
STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
```

- Point browser at:

```
http://your-system:2001/
```

- Sign-in
- Click “Internet Configurations” (if IBM i 6.1 or higher)
- Click “IBM Web Administration for i"

IBM Navigator for i



Welcome to the IBM Navigator for i

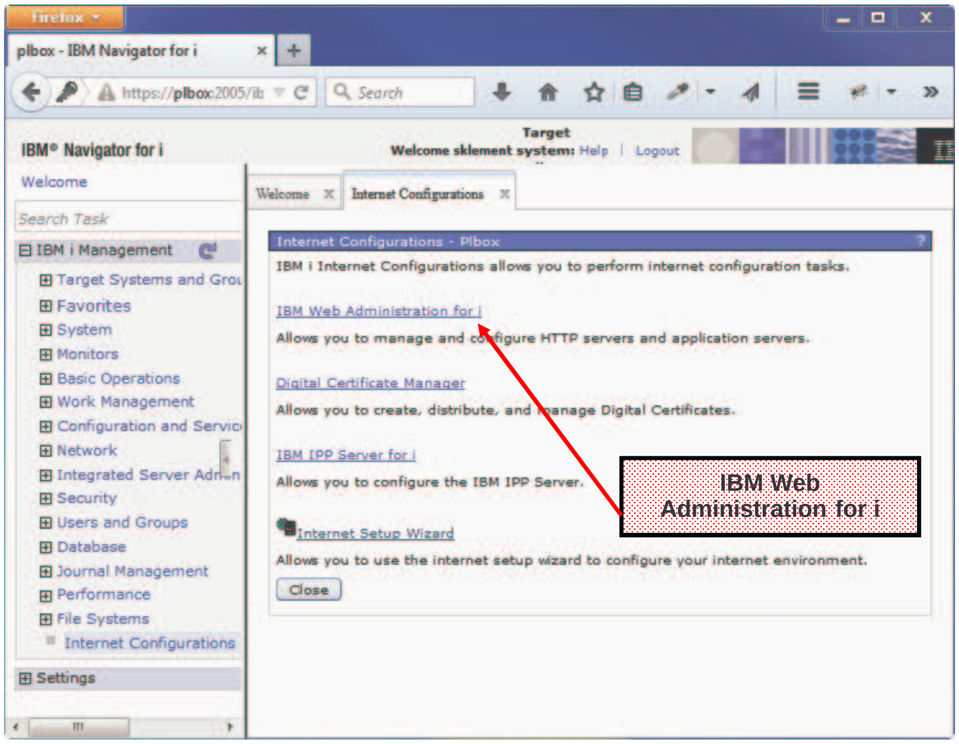
IBM Navigator for i provides an easy to use interface for the web-enabled IBM i management tasks, including all previous IBM i Navigator tasks on the web, and 2001 port tasks.

Expand IBM i Management in the left-hand navigation area to get started.

To see the previous version of the 2001 port tasks and where they are located now, click below.

Click "Internet Configurations"

Internet Configurations



IBM i Internet Configurations allows you to perform internet configuration tasks.

IBM Web Administration for i

Allows you to manage and configure HTTP servers and application servers.

Digital Certificate Manager

Allows you to create, distribute, and manage Digital Certificates.

IBM IPP Server for i

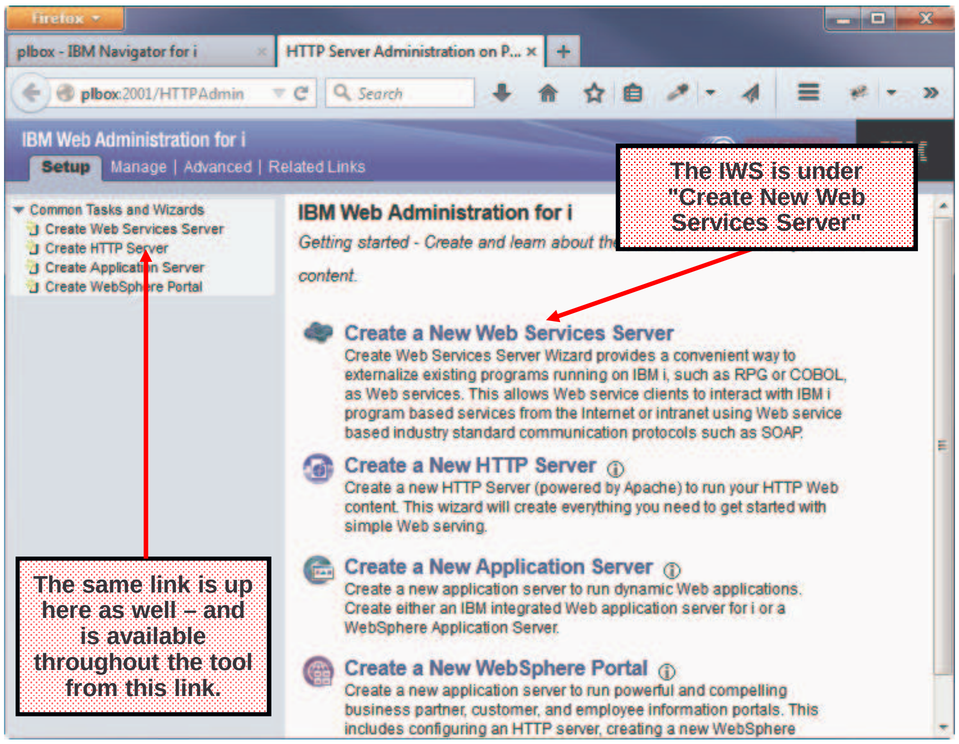
Allows you to configure the IBM IPP Server.

Internet Setup Wizard

Allows you to use the internet setup wizard to configure your internet environment.

IBM Web Administration for i

Web Administration for i


Firefox
plbox - IBM Navigator for i
HTTP Server Administration on P...
plbox:2001/HTTPAdmin
Search
IBM Web Administration for i
Setup | Manage | Advanced | Related Links
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal

IBM Web Administration for i

Getting started - Create and learn about the ... content.

Create a New Web Services Server
Create Web Services Server Wizard provides a convenient way to externalize existing programs running on IBM i, such as RPG or COBOL, as Web services. This allows Web service clients to interact with IBM i program based services from the Internet or intranet using Web service based industry standard communication protocols such as SOAP.

Create a New HTTP Server
Create a new HTTP Server (powered by Apache) to run your HTTP Web content. This wizard will create everything you need to get started with simple Web serving.

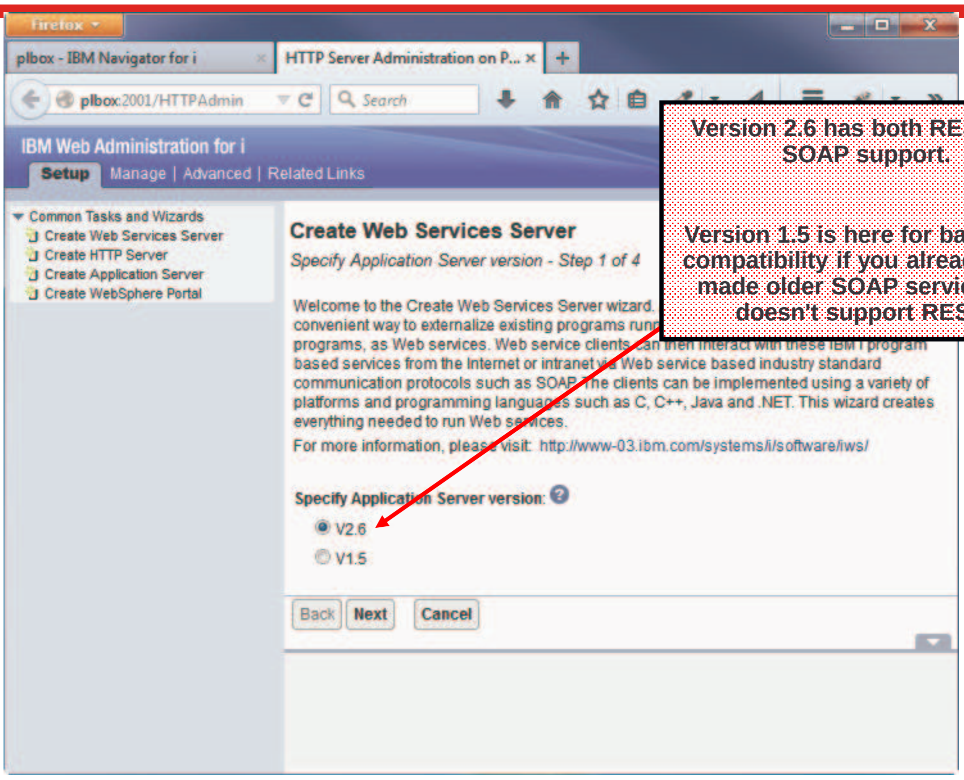
Create a New Application Server
Create a new application server to run dynamic Web applications. Create either an IBM integrated Web application server for i or a WebSphere Application Server.

Create a New WebSphere Portal
Create a new application server to run powerful and compelling business partner, customer, and employee information portals. This includes configuring an HTTP server, creating a new WebSphere

The IWS is under "Create New Web Services Server"

The same link is up here as well – and is available throughout the tool from this link.

Create IWS Server (1 of 4)


Firefox
plbox - IBM Navigator for i
HTTP Server Administration on P...
plbox:2001/HTTPAdmin
Search
IBM Web Administration for i
Setup | Manage | Advanced | Related Links
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Back
Next
Cancel

Create Web Services Server

Specify Application Server version - Step 1 of 4

Welcome to the Create Web Services Server wizard. ... convenient way to externalize existing programs run... programs, as Web services. Web service clients can then interact with these IBM i program based services from the Internet or intranet via Web service based industry standard communication protocols such as SOAP. The clients can be implemented using a variety of platforms and programming languages such as C, C++, Java and .NET. This wizard creates everything needed to run Web services.

For more information, please visit: http://www-03.ibm.com/systems/i/software/iws/

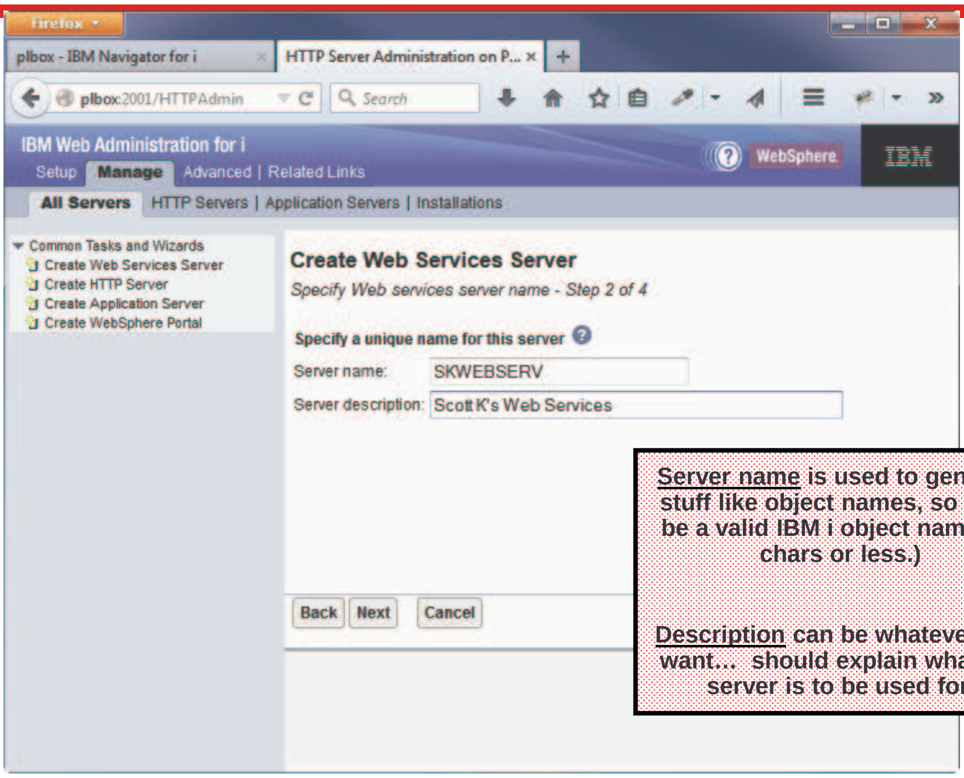
Specify Application Server version:

[✓ V2.6]
V1.5

Version 2.6 has both REST and SOAP support.

Version 1.5 is here for backward compatibility if you already have made older SOAP services. (It doesn't support REST.)

Create IWS Server (2 of 4)


Firefox
plbox - IBM Navigator for i
HTTP Server Administration on P...
plbox:2001/HTTPAdmin
Search
IBM Web Administration for i
Setup
Manage
Advanced | Related Links
WebSphere
IBM
All Servers
HTTP Servers | Application Servers | Installations
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Back
Next
Cancel

Create Web Services Server

Specify Web services server name - Step 2 of 4

Specify a unique name for this server

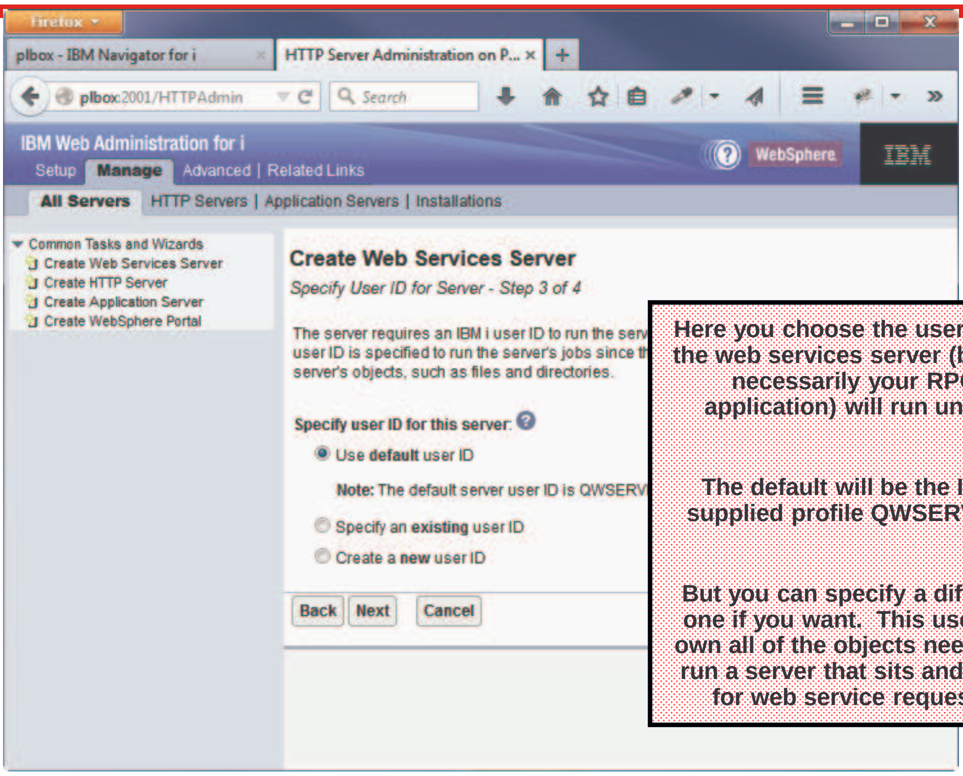
Server name: SKWEBSERV

Server description: Scott K's Web Services

<u>Server name</u> is used to generate stuff like object names, so must be a valid IBM i object name (10 chars or less.)

<u>Description</u> can be whatever you want… should explain what the server is to be used for.

Create IWS Server (3 of 4)


Firefox
plbox - IBM Navigator for i
HTTP Server Administration on P...
plbox:2001/HTTPAdmin
Search
IBM Web Administration for i
Setup
Manage
Advanced | Related Links
WebSphere
IBM
All Servers
HTTP Servers | Application Servers | Installations
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Back
Next
Cancel

Create Web Services Server

Specify User ID for Server - Step 3 of 4

The server requires an IBM i user ID to run the serv
user ID is specified to run the server's jobs since th
server's objects, such as files and directories.

Specify user ID for this server:

[✓ Use **default** user ID]

Note: The default server user ID is QWSERV

Specify an **existing** user ID

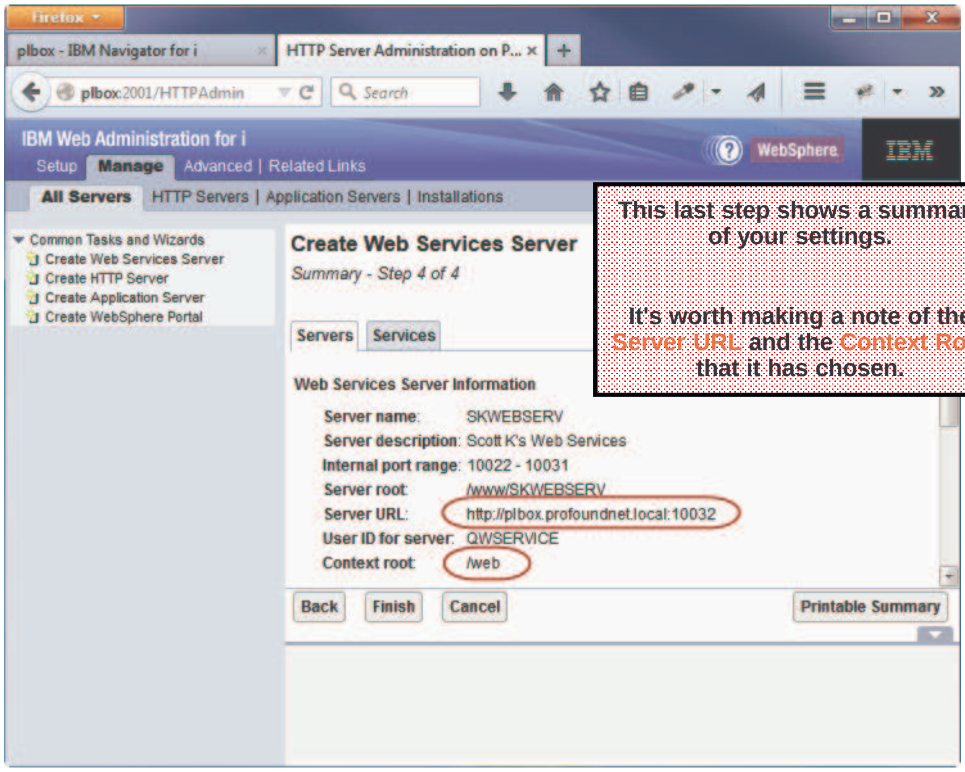
Create a **new** user ID

Here you choose the userid that the web services server (but not necessarily your RPG application) will run under.

The default will be the IBM-supplied profile QWSERVICE.

But you can specify a different one if you want. This user will own all of the objects needed to run a server that sits and waits for web service requests.

Create IWS Server (4 of 4)



Create Web Services Server

Summary - Step 4 of 4

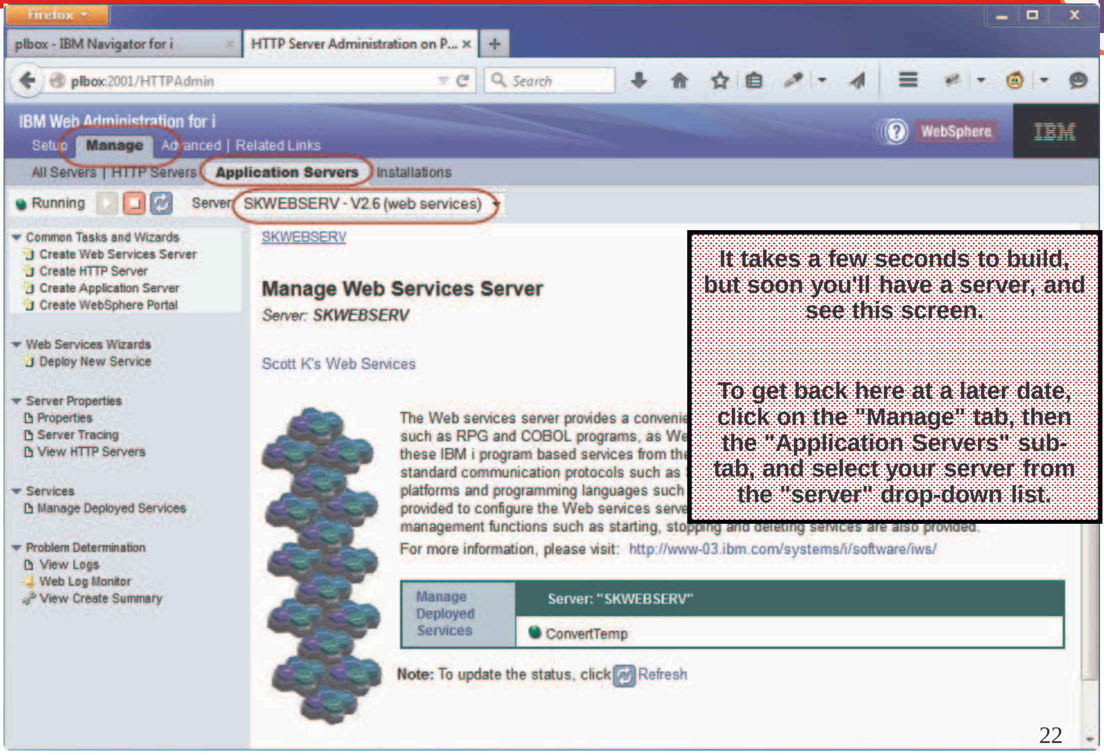
Web Services Server Information

Server name:	SKWEBSERV
Server description:	Scott K's Web Services
Internal port range:	10022 - 10031
Server root:	/www/SKWEBSERV
Server URL:	http://plbox.profoundnet.local:10032
User ID for server:	QWSERVICE
Context root:	/web

This last step shows a summary of your settings.

It's worth making a note of the Server URL and the Context Root that it has chosen.

We Now Have a Server!



Manage Web Services Server

Server: SKWEBSERV

Scott K's Web Services

The Web services server provides a convenie
such as RPG and COBOL programs, as We
these IBM i program based services from the
standard communication protocols such as
platforms and programming languages such
provided to configure the Web services serve
management functions such as starting, stopping and deleting services are also provided.

For more information, please visit: http://www-03.ibm.com/systems/i/software/iws/

Manage Deployed Services	Server: "SKWEBSERV"
	● ConvertTemp

Note: To update the status, click Refresh

It takes a few seconds to build, but soon you'll have a server, and see this screen.

To get back here at a later date, click on the "Manage" tab, then the "Application Servers" sub-tab, and select your server from the "server" drop-down list.

Now What?

Now that we have a web services server, we can add (or "deploy" is the official term) web services… i.e. programs/subprocedures that can be called as web services.

- One server can handle many services (programs/procedures)
- The same server can handle both REST and SOAP services (version 2.6+)
- IBM provides a "ConvertTemp" service as an example.

The "manage deployed services" button can be used to stop/start individual services as well as add/remove them.

SOAP Web Services

- Always XML (you could have a different "payload", but it'd be embedded in XML under the covers)
- SOAP is the XML format for the "parameters" when making a call
- URL and SoapAction HTTP header define the program to call.
- WSDL document describes the details (contains network info as well as an XML schema)

To understand Web Services Description Language (WSDL), think "how would you tell the world"?

- Documentation? (Word Doc, PDF, etc?)
- Sample programs?
- Or… info that can be used to generate programs?

WSDL Skeleton

```
<definitions>

    <types>
       definition of types........
    </types>

    <message>
       definition of a message....
    </message>

    <portType>
       definition of a port.......
    </portType>

    <binding>
       definition of a binding....
    </binding>

    <service>
       a logical grouping of ports...
    </service>

</definitions>
```

<types> = the data types that the web service uses.

<message> = the messages that are sent to and received from the web service.

<portType> = the operations (or, "programs/procedures" you can call for this web service.

<binding> = the network protocol used.

<service> = a grouping of ports. (Much like a service program contains a group of subprocedures.)

SOAP

SOAP = Simple Object Access Protocol

SOAP is an XML language that describes the parameters that you pass to the programs that you call. When calling a Web service, there are two SOAP documents -- an input document that you send to the program you're calling, and an output document that gets sent back to you.

"Simple" is perhaps a misnomer!

- Not as simple as RPG parameter lists.
- Not as simple as REST

SOAP Skeleton

Here's the skeleton of a SOAP message:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
       soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding" >

    <soap:Header>
       (optional) contains header info, like payment info or authentication info
         (crypto key, userid/password, etc)
    </soap:Header>

    <soap:Body>
       . . .
       Contains the parameter info. (Varies by application.)
       . . .
       <soap:Fault>
         (optional) error info.
       </soap:Fault>
       . . .
    </soap:Body>

</soap:Envelope>
```

Sample SOAP Documents

Some details removed for brevity….

Input Message

```
<soapenv:Envelope>
<soapenv:Body>
    <xsd:getcust>
       <xsd:args0>
          <xsd:CUSTNO>495</xsd:CUSTNO>
       </xsd:args0>
    </xsd:getcust>
  </soapenv:Body>
</soapenv:Envelope>
```

Output Message

```
<soapenv:Envelope>
   <soapenv:Body>
      <ns:getcustResponse>
         <ns:return>
            <ns:CITY>POMPANO BEACH</ns:CITY>
            <ns:NAME>ACME INC</ns:NAME>
            <ns:POSTAL>33064-2121</ns:POSTAL>
            <ns:STATE>FL</ns:STATE>
            <ns:STREET>123 MAIN STREET</ns:STREET>
         </ns:return>
      </ns:getcustResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

GETCUST RPG Program (1 of 2)

```
H DFTACTGRP(*NO) ACTGRP('SOAP') PGMINFO(*PCML: *MODULE)

FCUSTFILE  IF   E           K DISK    PREFIX('CUST.')

D CUST          E DS                  qualified
D                                     extname(CUSTFILE)

D GETCUST         PR                  ExtPgm('GETCUST')
D   CustNo                            like(Cust.Custno)
D   Name                              like(Cust.Name)
D   Street                            like(Cust.Street)
D   City                              like(Cust.City)
D   State                             like(Cust.State)
D   Postal                            like(Cust.Postal)
D GETCUST         PI
D   CustNo                            like(Cust.Custno)
D   Name                              like(Cust.Name)
D   Street                            like(Cust.Street)
D   City                              like(Cust.City)
D   State                             like(Cust.State)
D   Postal                            like(Cust.Postal)
```

PCML with parameter info will be embedded in the module and program objects.

This PREFIX causes the file to be read into the CUST data struct.

When there's no P-spec, the PR/PI acts the same as *ENTRY PLIST.

GETCUST RPG Program (2 of 2)

```
    /free
       chain CustNo CUSTFILE;
       if not %found;
          msgdta = 'Customer not found.';
          QMHSNDPM( 'CPF9897': 'QCPFMSG   *LIBL'
                  : msgdta: %len(msgdta): '*ESCAPE'
                  : '*PGMBDY': 1: MsgKey: err );
       else;
          Custno = Cust.Custno;
          Name   = Cust.name;
          Street = Cust.Street;
          City   = Cust.City;
          State  = Cust.State;
          Postal = Cust.Postal;
       endif;
       *inlr = *on;
    /end-free
```

This API is equivalent to the CL SNDPGMMSG command, and causes my program to end with an exception ("halt")

When there are no errors, I simply return my output via the parameter list. IWS takes care of the XML for me!

PCML so IWS Knows Our Parameters

Our GETCUST example gets input and output as normal parameters. To use these with IWS, we need to tell IWS what these parameters are. This is done with yet another XML document.

PCML = Program Call Markup Language

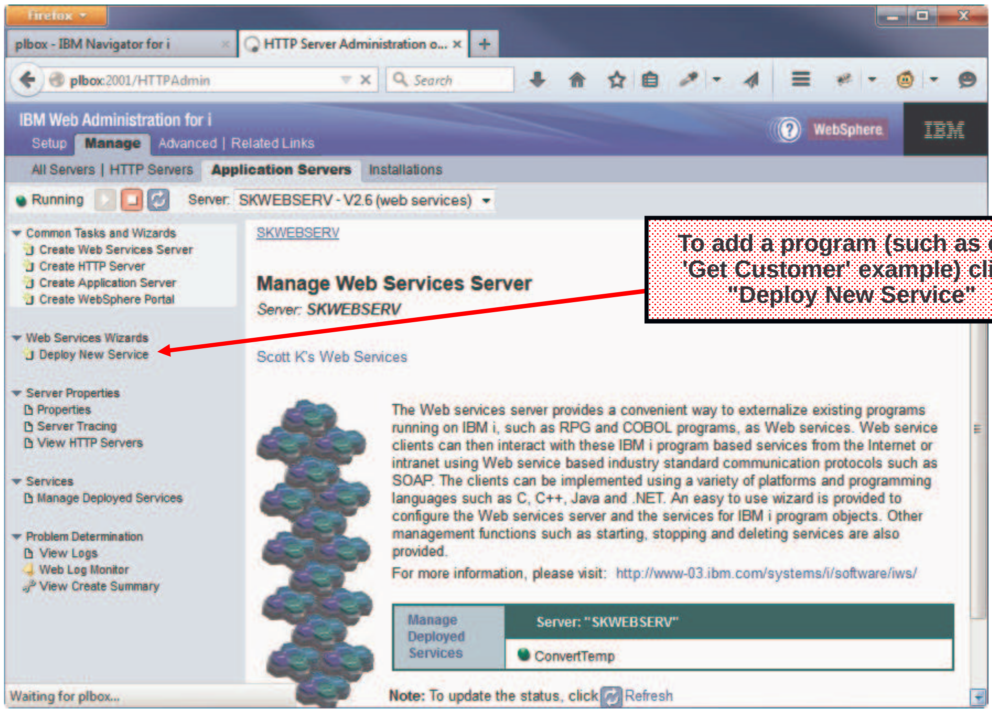
- A flavor of XML that describes a program's (or *SRVPGM's) parameters.
- Can be generated for you by the RPG compiler, and stored in the IFS:

```
CRTBNDRPG PGM(xyz) SRCFILE(QRPGLESRC)
          PGMINFO(*PCML)
          INFOSTMF('/path/to/myfile.pcml')
```

- Or can be embedded into the module/program objects themselves, with an H-spec:

```
H PGMINFO(*PCML:*MODULE)
```

GETCUST as SOAP Service



SKWEBSERV

Manage Web Services Server

Server: SKWEBSERV

Scott K's Web Services

The Web services server provides a convenient way to externalize existing programs running on IBM i, such as RPG and COBOL programs, as Web services. Web service clients can then interact with these IBM i program based services from the Internet or intranet using Web service based industry standard communication protocols such as SOAP. The clients can be implemented using a variety of platforms and programming languages such as C, C++, Java and .NET. An easy to use wizard is provided to configure the Web services server and the services for IBM i program objects. Other management functions such as starting, stopping and deleting services are also provided.

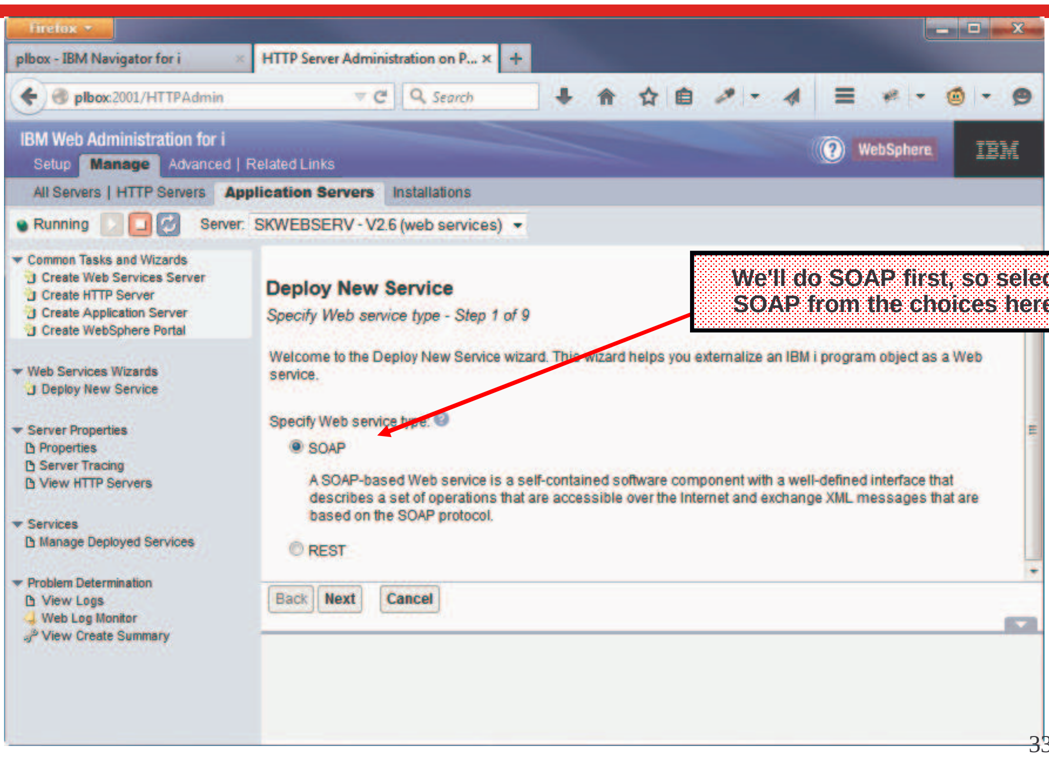
For more information, please visit: http://www-03.ibm.com/systems/i/software/iws/

Manage Deployed Services	Server: "SKWEBSERV"
	● ConvertTemp

Note: To update the status, click Refresh

To add a program (such as our 'Get Customer' example) click "Deploy New Service"

SOAP Example (1 of 9)



Deploy New Service

Specify Web service type - Step 1 of 9

Welcome to the Deploy New Service wizard. This wizard helps you externalize an IBM i program object as a Web service.

Specify Web service type:

[✓ SOAP]

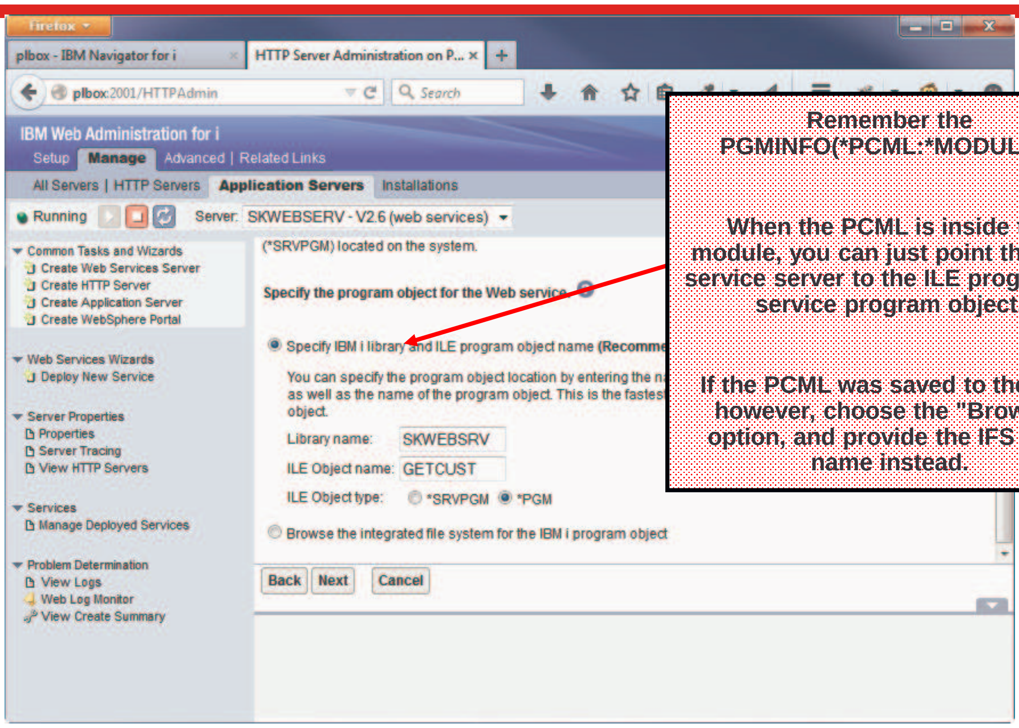
A SOAP-based Web service is a self-contained software component with a well-defined interface that describes a set of operations that are accessible over the Internet and exchange XML messages that are based on the SOAP protocol.

REST

Server: SKWEBSERV - V2.6 (web services)

We'll do SOAP first, so select SOAP from the choices here.

SOAP Example (2 of 9)



(*SRVPGM) located on the system.

Specify the program object for the Web service

[✓ Specify IBM i library and ILE program object name (Recomme]

You can specify the program object location by entering the n… as well as the name of the program object. This is the fastes… object.

Library name: SKWEBSRV

ILE Object name: GETCUST

ILE Object type: *SRVPGM [✓ *PGM]

Browse the integrated file system for the IBM i program object

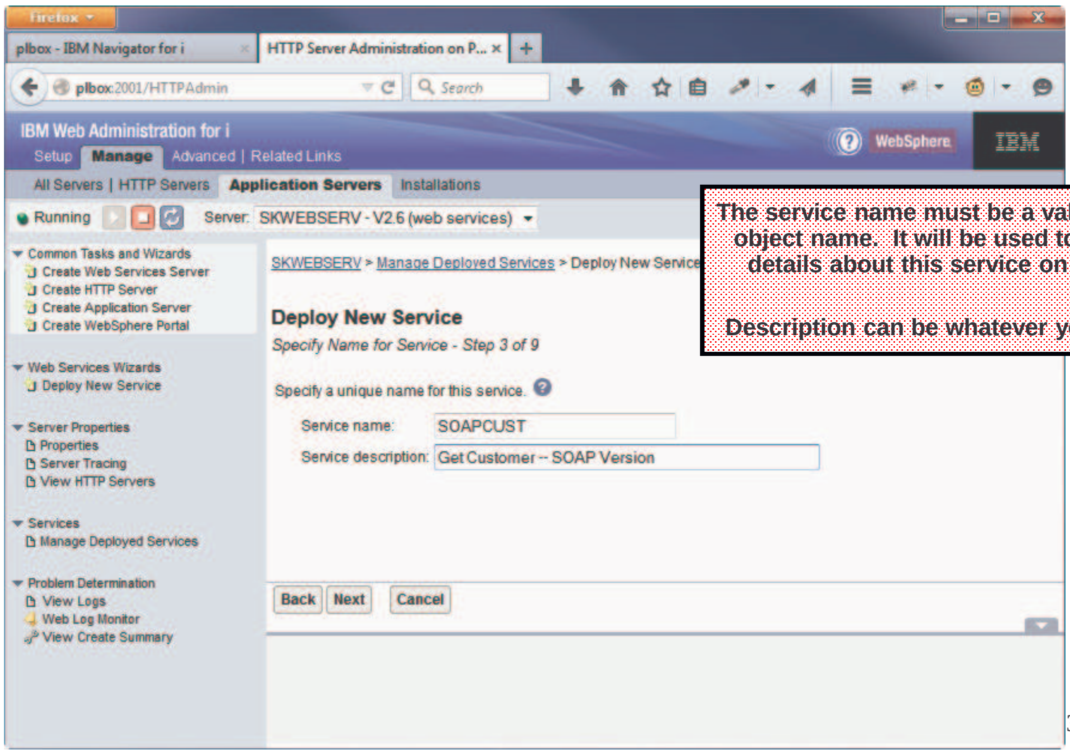
Server: SKWEBSERV - V2.6 (web services)

Remember the PGMINFO(*PCML:*MODULE)?

When the PCML is inside the module, you can just point the web service server to the ILE program or service program object.

If the PCML was saved to the IFS, however, choose the "Browse" option, and provide the IFS path name instead.

SOAP Example (3 of 9)



Server: SKWEBSERV - V2.6 (web services)

SKWEBSERV > Manage Deployed Services > Deploy New Service

Deploy New Service

Specify Name for Service - Step 3 of 9

Specify a unique name for this service.

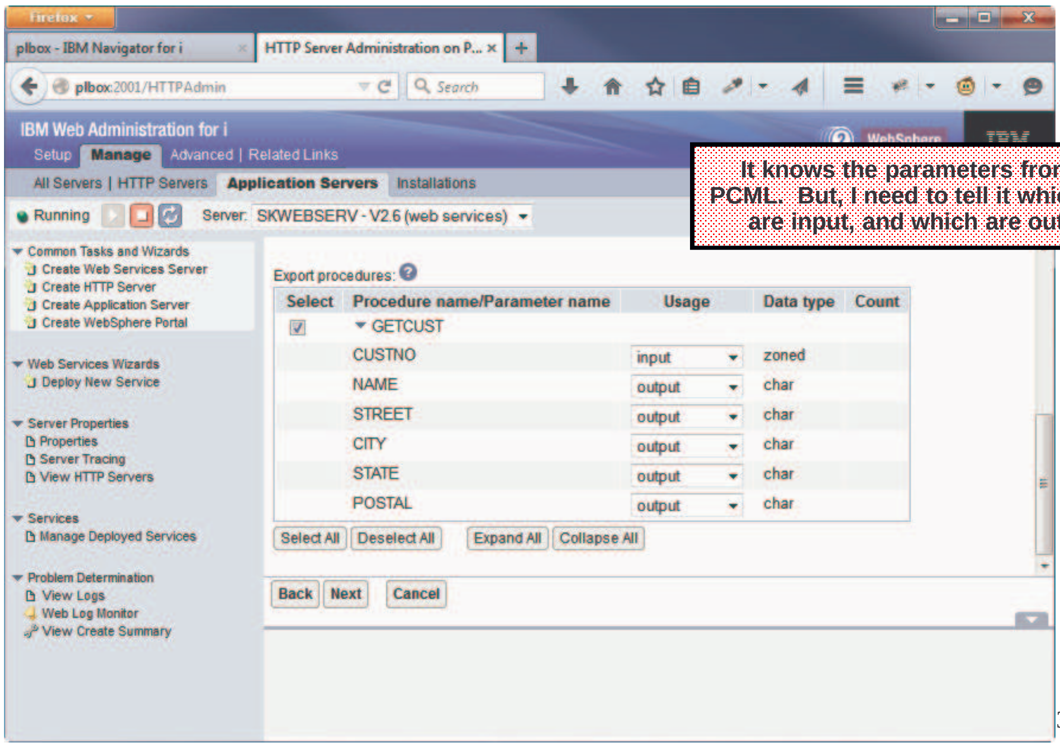
Service name: SOAPCUST

Service description: Get Customer -- SOAP Version

The service name must be a valid IBM i object name. It will be used to store details about this service on disk.

Description can be whatever you like.

SOAP Example (4 of 9)



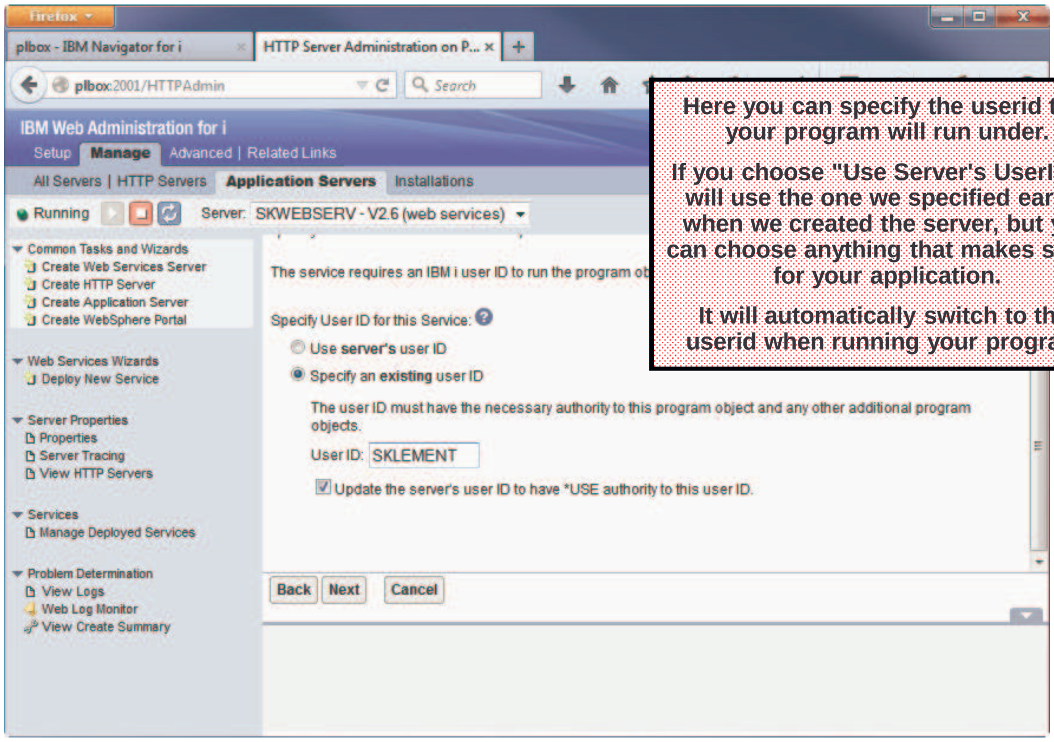
Server: SKWEBSERV - V2.6 (web services)

Export procedures:

Select	Procedure name/Parameter name	Usage	Data type	Count
✓	GETCUST			
	CUSTNO	input	zoned	
	NAME	output	char	
	STREET	output	char	
	CITY	output	char	
	STATE	output	char	
	POSTAL	output	char	

It knows the parameters from the PCML. But, I need to tell it which ones are input, and which are output.

SOAP Example (5 of 9)


plbox - IBM Navigator for i | HTTP Server Administration on P...

plbox:2001/HTTPAdmin

IBM Web Administration for i

Setup | Manage | Advanced | Related Links

All Servers | HTTP Servers | Application Servers | Installations

Running — Server: SKWEBSERV - V2.6 (web services)

The service requires an IBM i user ID to run the program ob

Specify User ID for this Service:

( ) Use server's user ID

(●) Specify an existing user ID

The user ID must have the necessary authority to this program object and any other additional program objects.

User ID: SKLEMENT

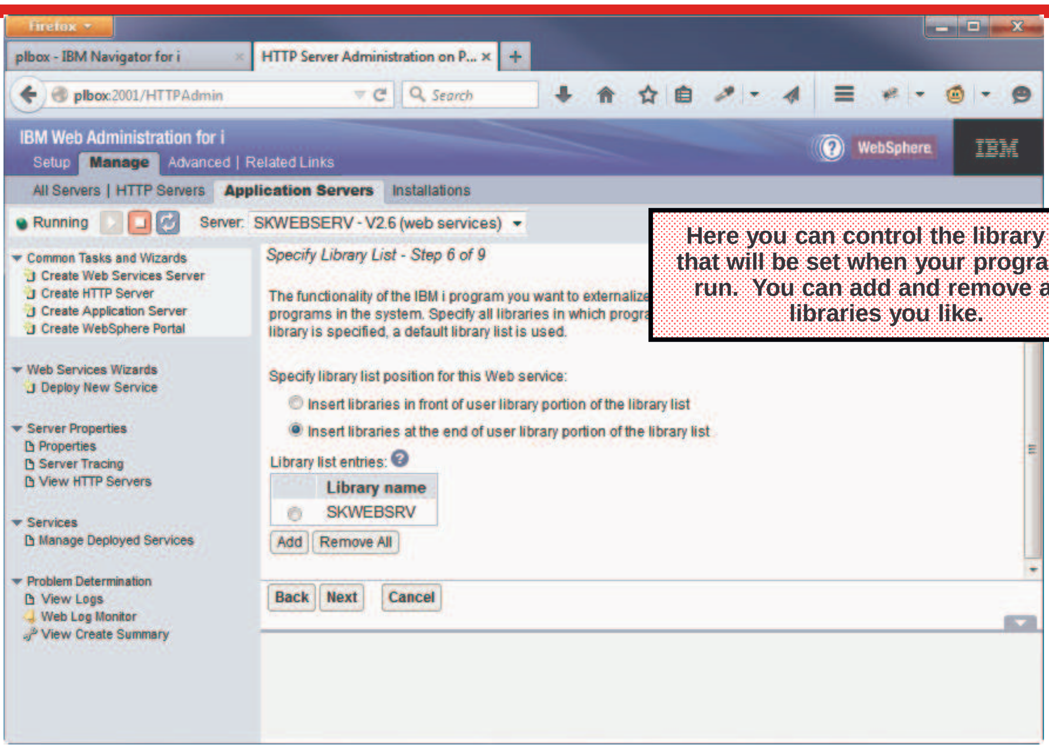
[✓] Update the server's user ID to have *USE authority to this user ID.

Here you can specify the userid that your program will run under.

If you choose "Use Server's UserID" it will use the one we specified earlier when we created the server, but you can choose anything that makes sense for your application.

It will automatically switch to this userid when running your program.

SOAP Example (6 of 9)


plbox - IBM Navigator for i | HTTP Server Administration on P...

plbox:2001/HTTPAdmin

IBM Web Administration for i

Setup | Manage | Advanced | Related Links

All Servers | HTTP Servers | Application Servers | Installations

Running — Server: SKWEBSERV - V2.6 (web services)

Specify Library List - Step 6 of 9

The functionality of the IBM i program you want to externalize programs in the system. Specify all libraries in which progra library is specified, a default library list is used.

Specify library list position for this Web service:

( ) Insert libraries in front of user library portion of the library list

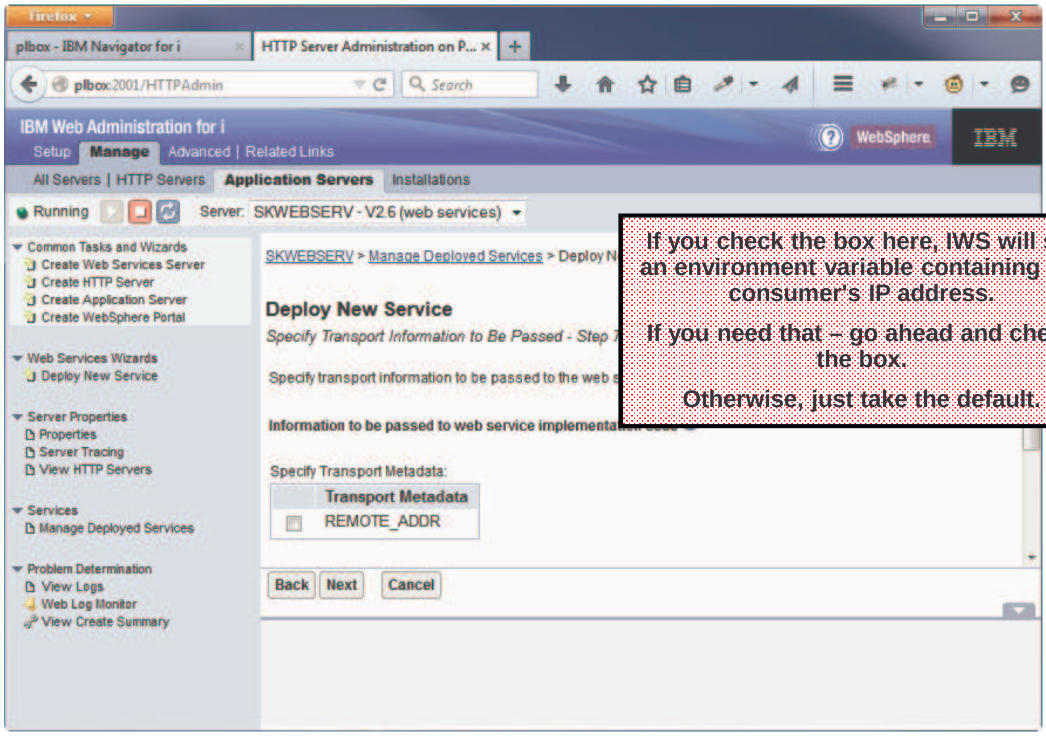
(●) Insert libraries at the end of user library portion of the library list

Library list entries:

	Library name
○	SKWEBSRV

Here you can control the library list that will be set when your program is run. You can add and remove any libraries you like.

SOAP Example (7 of 9)



IBM Web Administration for i

Setup | Manage | Advanced | Related Links

All Servers | HTTP Servers | Application Servers | Installations

Running Server: SKWEBSERV - V2.6 (web services)

SKWEBSERV > Manage Deployed Services > Deploy N

Deploy New Service

Specify Transport Information to Be Passed - Step 7

Specify transport information to be passed to the web s

Information to be passed to web service implementat

Specify Transport Metadata:

	Transport Metadata
☐	REMOTE_ADDR

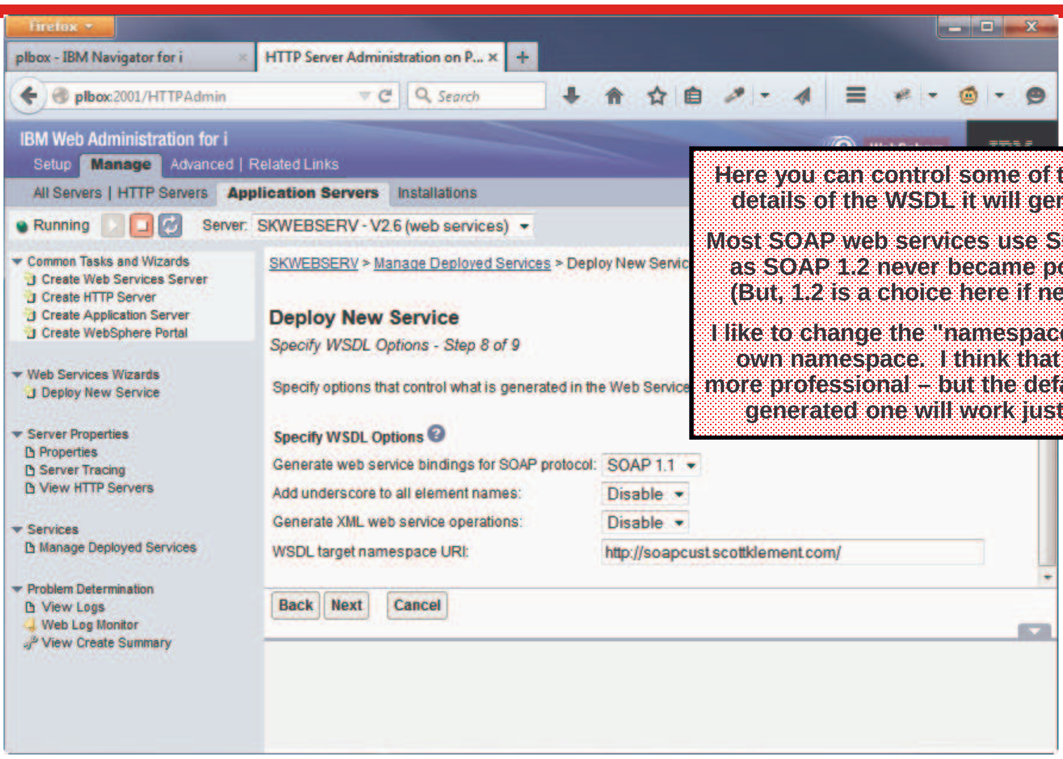
Back | Next | Cancel

If you check the box here, IWS will set an environment variable containing the consumer's IP address.

If you need that – go ahead and check the box.

Otherwise, just take the default.

SOAP Example (8 of 9)



IBM Web Administration for i

Setup | Manage | Advanced | Related Links

All Servers | HTTP Servers | Application Servers | Installations

Running Server: SKWEBSERV - V2.6 (web services)

SKWEBSERV > Manage Deployed Services > Deploy New Servic

Deploy New Service

Specify WSDL Options - Step 8 of 9

Specify options that control what is generated in the Web Service

Specify WSDL Options

Generate web service bindings for SOAP protocol:	SOAP 1.1
Add underscore to all element names:	Disable
Generate XML web service operations:	Disable
WSDL target namespace URI:	http://soapcust.scottklement.com/

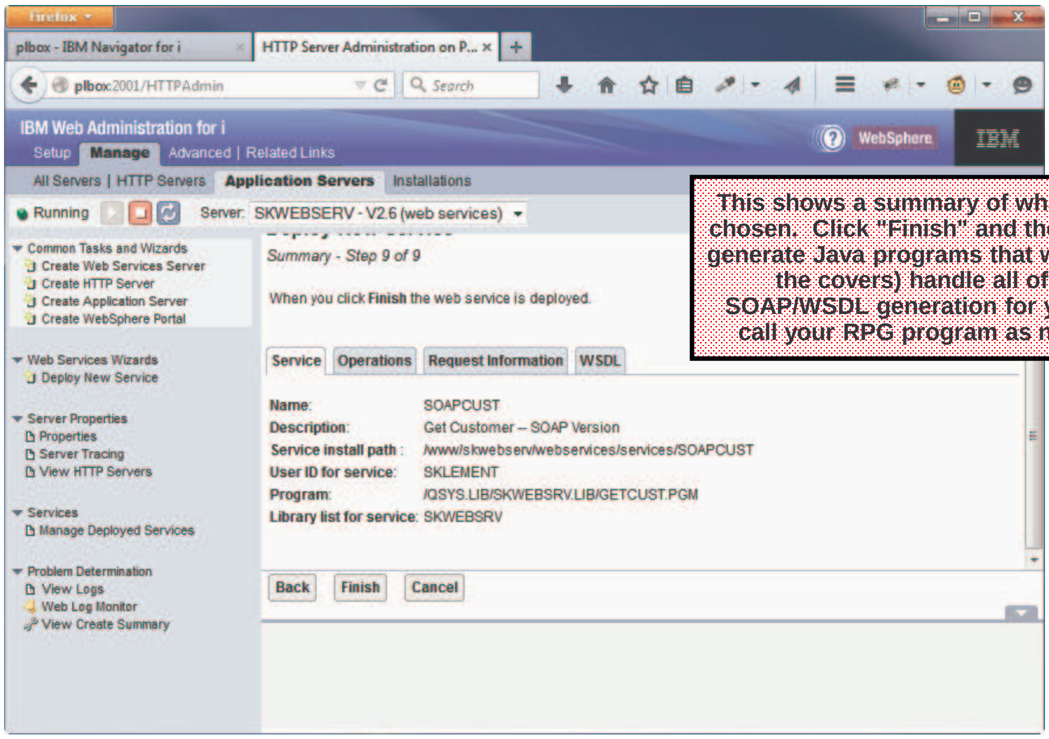
Back | Next | Cancel

Here you can control some of the finer details of the WSDL it will generate.

Most SOAP web services use SOAP 1.1, as SOAP 1.2 never became popular. (But, 1.2 is a choice here if needed.)

I like to change the "namespace" to my own namespace. I think that looks more professional – but the default IBM-generated one will work just fine.

SOAP Example (9 of 9)


Firefox
plbox - IBM Navigator for i
HTTP Server Administration on P...
plbox:2001/HTTPAdmin
IBM Web Administration for i
Setup | Manage | Advanced | Related Links
All Servers | HTTP Servers | Application Servers | Installations
Running
Server: SKWEBSERV - V2.6 (web services)
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Web Services Wizards
Deploy New Service
Server Properties
Properties
Server Tracing
View HTTP Servers
Services
Manage Deployed Services
Problem Determination
View Logs
Web Log Monitor
View Create Summary

Summary - Step 9 of 9

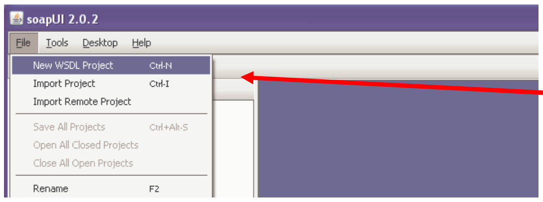
When you click **Finish** the web service is deployed.

Service | Operations | Request Information | WSDL

Name:	SOAPCUST
Description:	Get Customer – SOAP Version
Service install path:	/www/skwebserv/webservices/services/SOAPCUST
User ID for service:	SKLEMENT
Program:	/QSYS.LIB/SKWEBSRV.LIB/GETCUST.PGM
Library list for service:	SKWEBSRV

This shows a summary of what you've chosen. Click "Finish" and the IWS will generate Java programs that will (under the covers) handle all of the SOAP/WSDL generation for you, and call your RPG program as needed.

Testing SOAP with SoapUI (1 of 4)



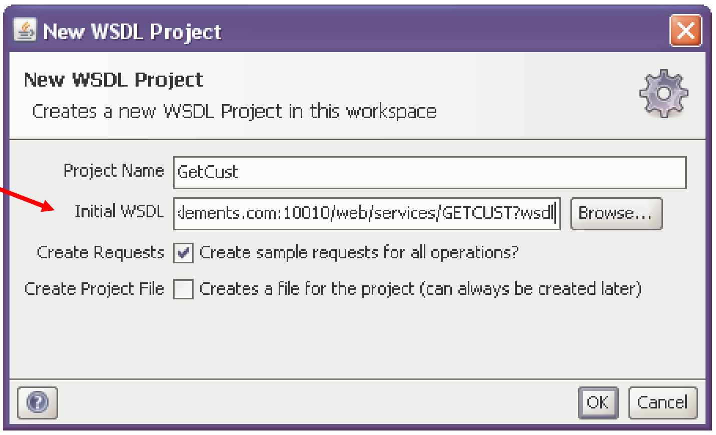
Step 1:

Click File -> New Project

(some versions say "WSDL project", others say "SoapUI project. They're the same.)

Step 2:

Paste in URL to WSDL (from the "View Service Definition" link) into the Initial WSDL blank.



New WSDL Project

Creates a new WSDL Project in this workspace

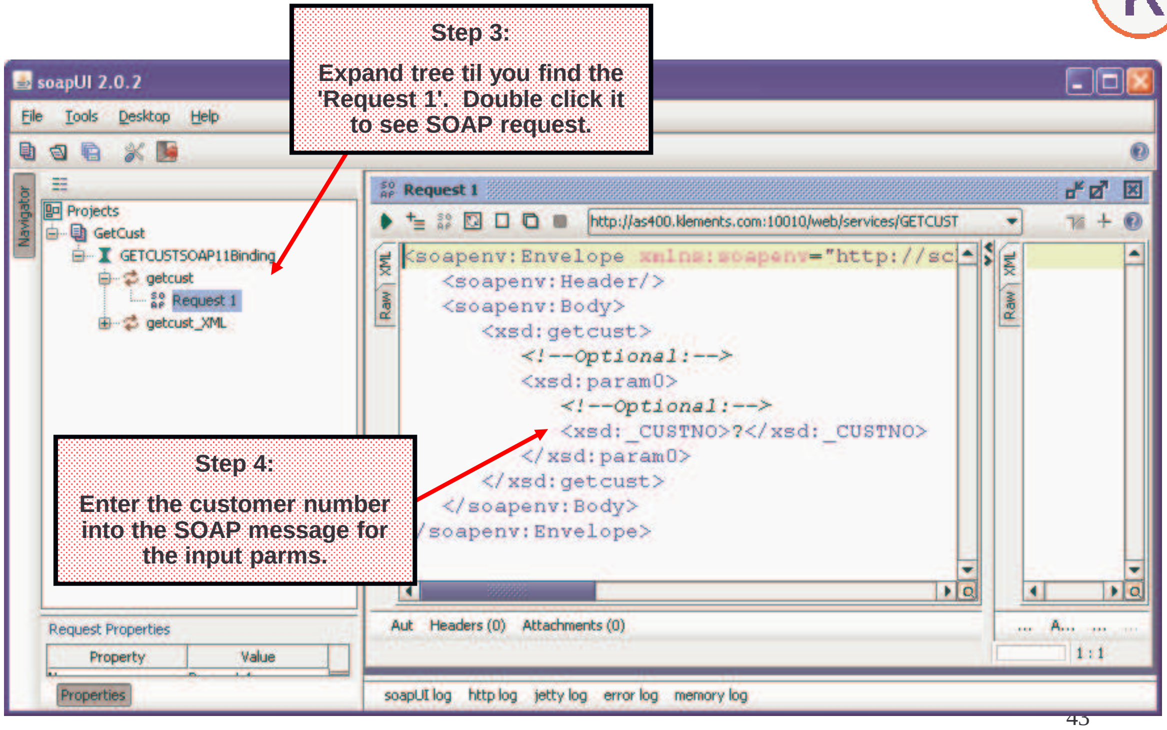
Project Name: GetCust

Initial WSDL: klements.com:10010/web/services/GETCUST?wsdl

Create Requests: [✓] Create sample requests for all operations?

Create Project File: [ ] Creates a file for the project (can always be created later)

Testing SOAP with SoapUI (2 of 4)


Step 3:

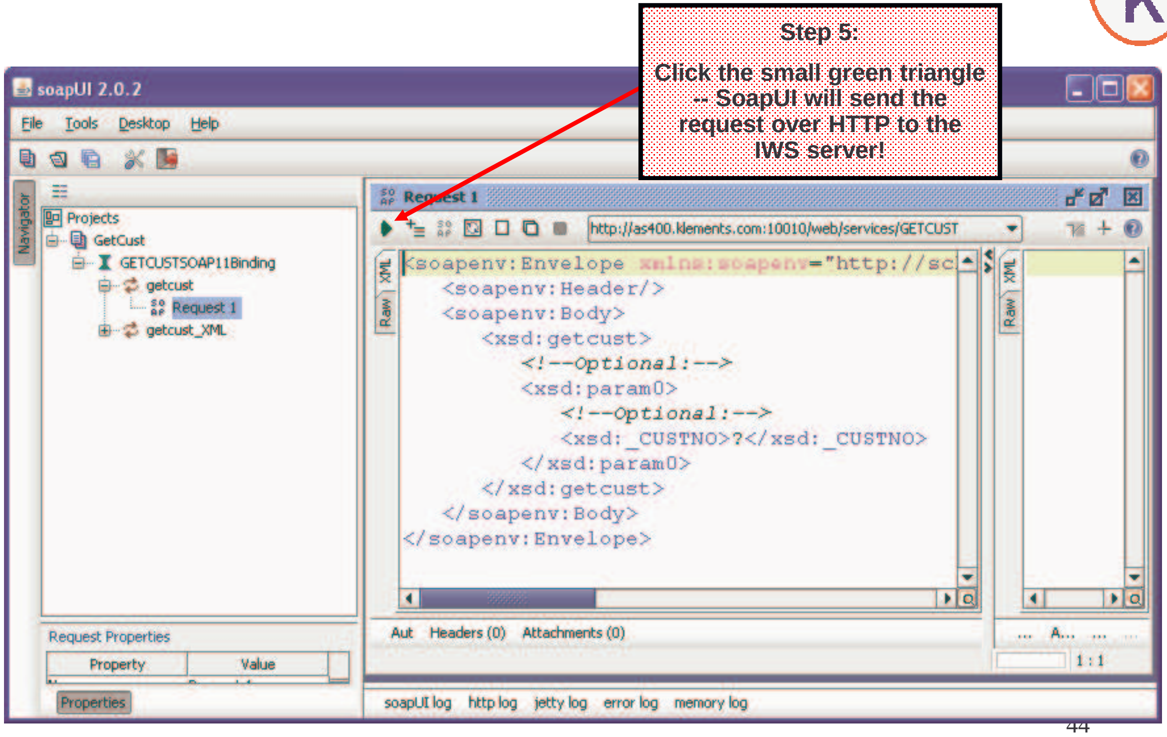
Expand tree til you find the 'Request 1'. Double click it to see SOAP request.

Step 4:

Enter the customer number into the SOAP message for the input parms.

```
<soapenv:Envelope xmlns:soapenv="http://sc
   <soapenv:Header/>
   <soapenv:Body>
      <xsd:getcust>
         <!--Optional:-->
         <xsd:param0>
            <!--Optional:-->
            <xsd:_CUSTNO>?</xsd:_CUSTNO>
         </xsd:param0>
      </xsd:getcust>
   </soapenv:Body>
</soapenv:Envelope>
```

Testing SOAP with SoapUI (3 of 4)


Step 5:

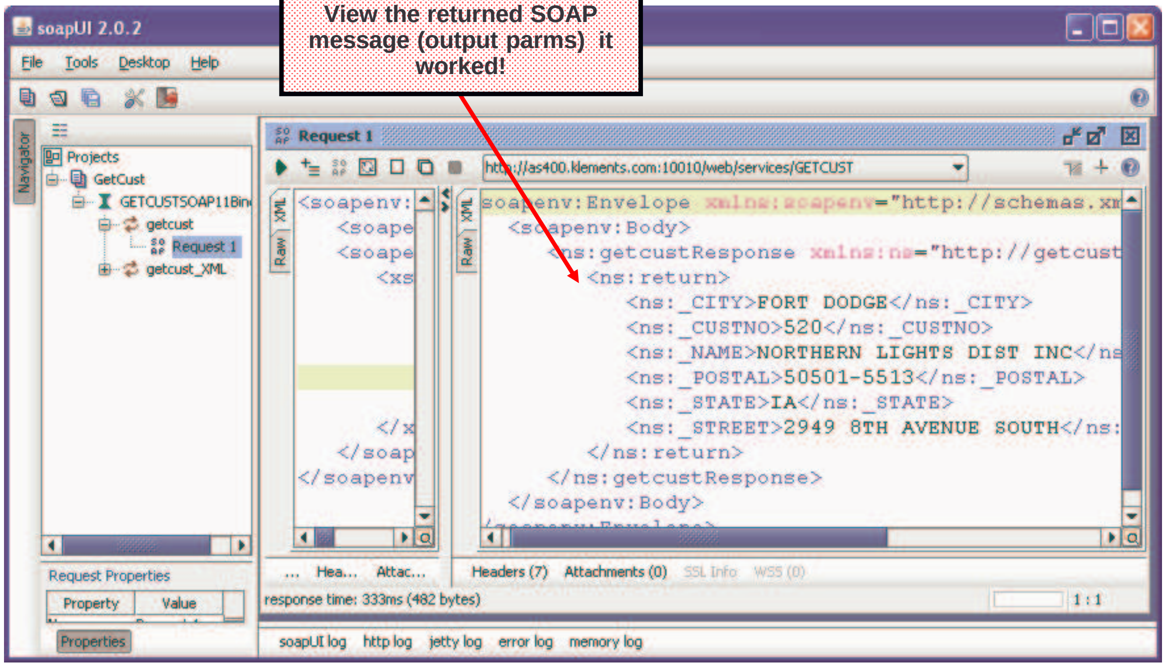
Click the small green triangle -- SoapUI will send the request over HTTP to the IWS server!

```
<soapenv:Envelope xmlns:soapenv="http://sc
   <soapenv:Header/>
   <soapenv:Body>
      <xsd:getcust>
         <!--Optional:-->
         <xsd:param0>
            <!--Optional:-->
            <xsd:_CUSTNO>?</xsd:_CUSTNO>
         </xsd:param0>
      </xsd:getcust>
   </soapenv:Body>
</soapenv:Envelope>
```

Testing SOAP with SoapUI (4 of 4)

Step 6:

View the returned SOAP message (output parms) it worked!



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xm
  <soapenv:Body>
    <ns:getcustResponse xmlns:ns="http://getcust
      <ns:return>
        <ns:_CITY>FORT DODGE</ns:_CITY>
        <ns:_CUSTNO>520</ns:_CUSTNO>
        <ns:_NAME>NORTHERN LIGHTS DIST INC</ns
        <ns:_POSTAL>50501-5513</ns:_POSTAL>
        <ns:_STATE>IA</ns:_STATE>
        <ns:_STREET>2949 8TH AVENUE SOUTH</ns:
      </ns:return>
    </ns:getcustResponse>
  </soapenv:Body>
```

response time: 333ms (482 bytes)

After SOAP, I Need a REST

Remember that REST (sometimes called 'RESTful') web services differ from SOAP in that:

- the URL points to a "noun" (or "resource")
- the HTTP method specifies a "verb" like GET, POST, PUT or DELETE. (Similar to a database Create, Read, Update, Delete…)
- REST sounds nicer than CRUD, haha.

IWS structures the URL like this:

```
http://address:port/context-root/root-resource/path-template
```

- context-root = Distinguishes from other servers. The default context-root is /web/services, but you can change this in the server properties.
- root-resource = identifies the type of resource (or "noun") we're working with. In our example, we'll use "/cust" to identify a customer. The IWS will also use this to determine which program to run.
- path-template = identifies the variables/parameters that distinguish this noun from others. In our example, it'll be the customer number.

Example REST Input

For our example, we will use this URL:

```
http://address:port/web/services/cust/495
```

Our URL will represent a customer record. Then we can:

- GET <url> the customer to see the address.
- potentially POST <url> the customer to create a new customer record
- potentially PUT <url> the customer to update an existing customer record
- potentially DELETE <url> to remove the customer record.

Though, in this particular example, our requirements are only to retrieve customer details, so we won't do all four possible verbs, we'll only do GET.

That means in IWS terminology:

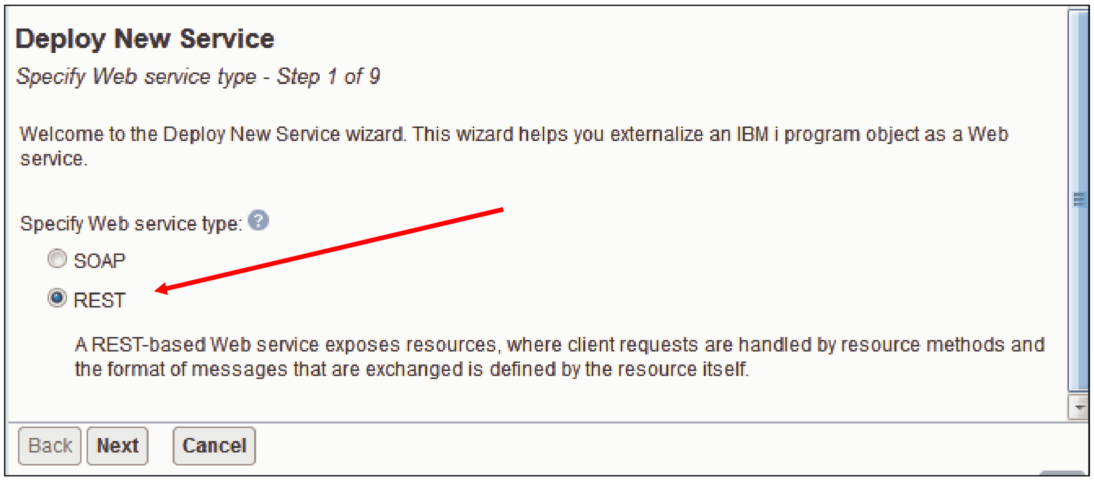
- /web/services is the context root.
- /cust is the root resource (and will point to our GETCUST program)
- /495 (or any other customer number) is the path template.

With that in mind, we're off to see the wizard… the wonderful wizard of REST.

REST Wizard (1 of 9)

Now I'd like to do the same web service as REST instead of SOAP. (The IWS also supports REST in the latest versions.)

To do that, I'll click 'Deploy New Service' again, this time choosing REST.



Deploy New Service

Specify Web service type - Step 1 of 9

Welcome to the Deploy New Service wizard. This wizard helps you externalize an IBM i program object as a Web service.

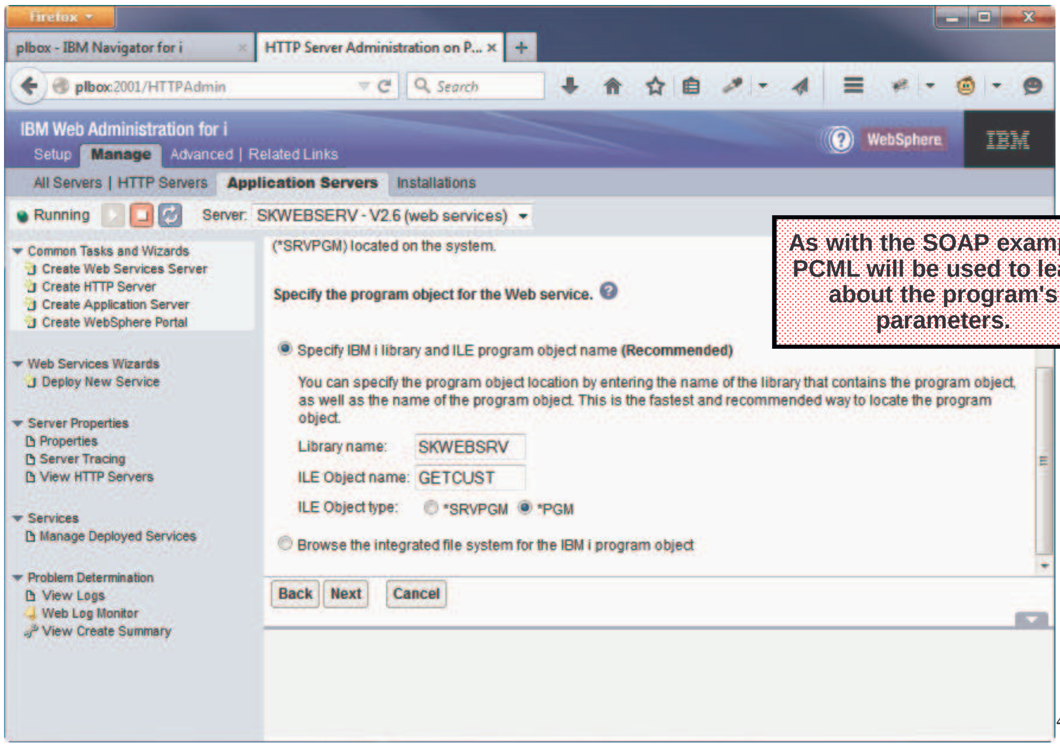
Specify Web service type:

○ SOAP

● REST

A REST-based Web service exposes resources, where client requests are handled by resource methods and the format of messages that are exchanged is defined by the resource itself.

REST Wizard (2 of 9)


Firefox
plbox - IBM Navigator for i
HTTP Server Administration on P...
plbox:2001/HTTPAdmin
IBM Web Administration for i
Setup | Manage | Advanced | Related Links
All Servers | HTTP Servers | Application Servers | Installations
Running
Server: SKWEBSERV - V2.6 (web services)

Common Tasks and Wizards
- Create Web Services Server
- Create HTTP Server
- Create Application Server
- Create WebSphere Portal

Web Services Wizards
- Deploy New Service

Server Properties
- Properties
- Server Tracing
- View HTTP Servers

Services
- Manage Deployed Services

Problem Determination
- View Logs
- Web Log Monitor
- View Create Summary

(*SRVPGM) located on the system.

Specify the program object for the Web service.

(●) Specify IBM i library and ILE program object name **(Recommended)**

You can specify the program object location by entering the name of the library that contains the program object, as well as the name of the program object. This is the fastest and recommended way to locate the program object.

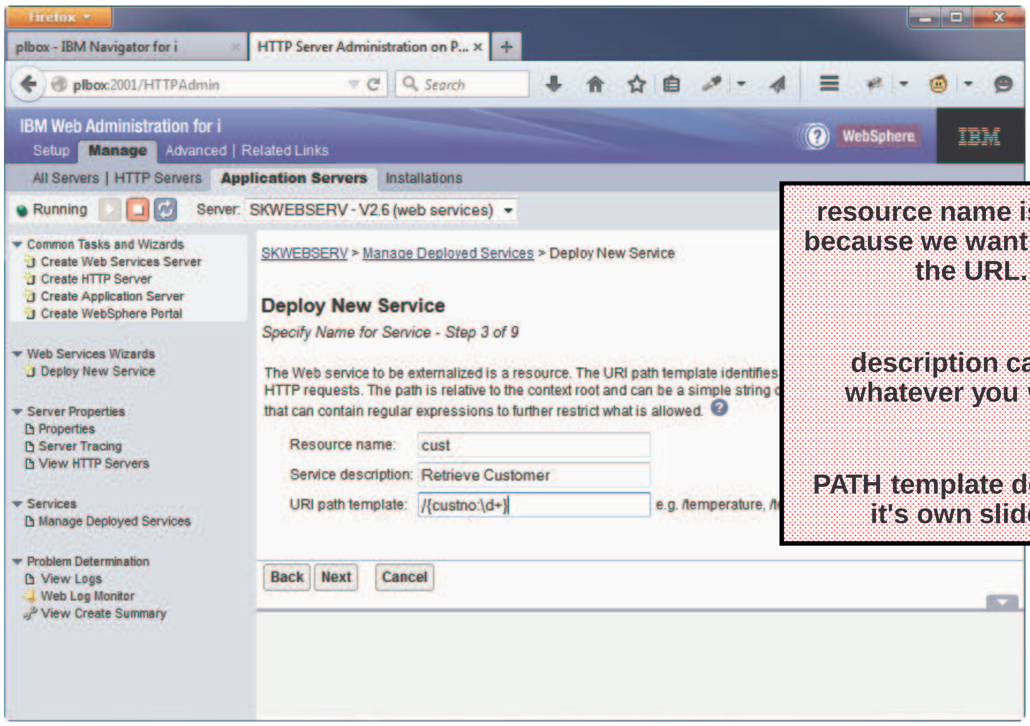
Library name:	SKWEBSRV
ILE Object name:	GETCUST
ILE Object type:	○ *SRVPGM ● *PGM

( ) Browse the integrated file system for the IBM i program object

Back | Next | Cancel

As with the SOAP example, PCML will be used to learn about the program's parameters.

REST Wizard (3 of 9)


Firefox
plbox - IBM Navigator for i
HTTP Server Administration on P...
plbox:2001/HTTPAdmin
IBM Web Administration for i
Setup | Manage | Advanced | Related Links
All Servers | HTTP Servers | Application Servers | Installations
Running
Server: SKWEBSERV - V2.6 (web services)

Common Tasks and Wizards
- Create Web Services Server
- Create HTTP Server
- Create Application Server
- Create WebSphere Portal

Web Services Wizards
- Deploy New Service

Server Properties
- Properties
- Server Tracing
- View HTTP Servers

Services
- Manage Deployed Services

Problem Determination
- View Logs
- Web Log Monitor
- View Create Summary

SKWEBSERV > Manage Deployed Services > Deploy New Service

Deploy New Service

Specify Name for Service - Step 3 of 9

The Web service to be externalized is a resource. The URI path template identifies HTTP requests. The path is relative to the context root and can be a simple string that can contain regular expressions to further restrict what is allowed.

Resource name:	cust
Service description:	Retrieve Customer
URI path template:	/{custno:\d+}

e.g. /temperature, /

Back | Next | Cancel

resource name is 'cust', because we want /cust/ in the URL.

description can be whatever you want.

PATH template deserves it's own slide ☺

Path Templates

You can make your URL as sophisticated as you like with a REST service. For example:

- Maybe there are multiple path variables separated by slashes
- Maybe they allow only numeric values
- Maybe they allow only letters, or only uppercase letters, or only lowercase, or both letters and numbers
- maybe they have to have certain punctuation, like slashes in a date, or dashes in a phone number.

Path templates are how you configure all of that. They have a syntax like:

```
{ identifier : regular expression }
```

- The identifier will be used later to map the variable into a program's parameter.
- The regular expression is used to tell IWS what is allowed in the parameter

Path Template Examples

For our example, we want /495 (or any other customer number) in the URL, so we do:

/{custno:\d+} identifier=custno, and regular expression \d+ means
\d = any digit, + = one or more

As a more sophisticated example, consider a web service that returns inventory in a particular warehouse location. The path template might identify a warehouse location in this syntax

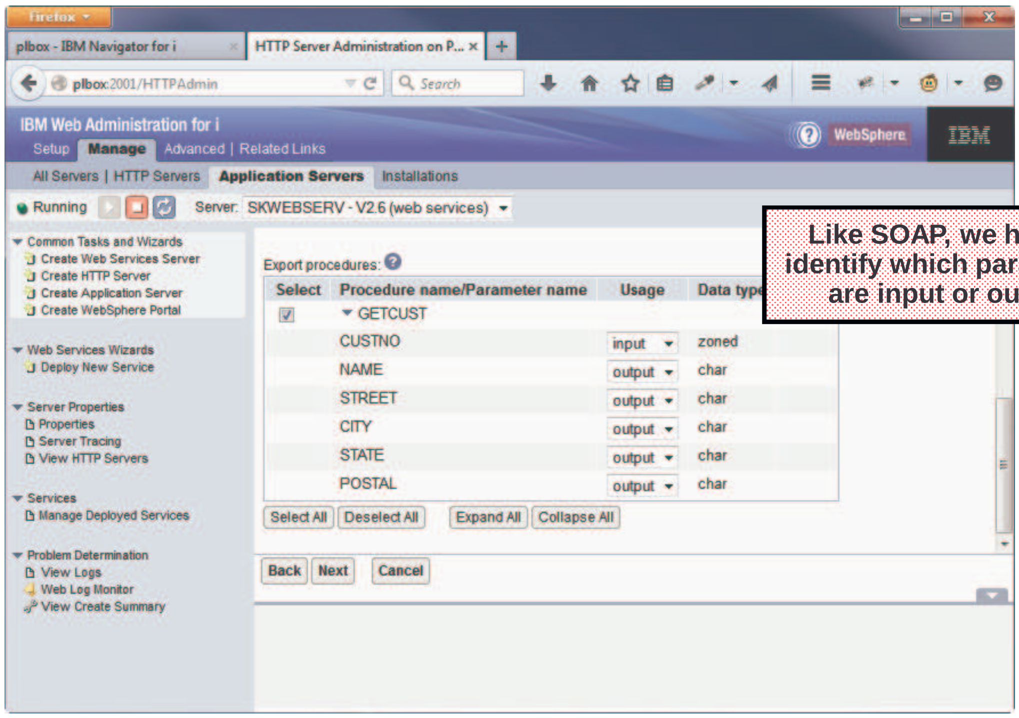
```
/Milwaukee/202/Freezer1/B/12/C
```

These identify City, Building, Room, Aisle, Slot and Shelf. The path template might be
/{city:\w+}/{bldg:\d+}/{room:\w+}/{aisle:[A-Z]}/{slot:\d\d}/{shelf:[A-E]}

\w+ = one or more of A-Z, a-z or 0-9 characters.
Aisle is only one letter, but can be A-Z (capital)
slot is always a two-digit number, from 00-99, \d\d means two numeric digits
Shelf is always capital letters A,B,C,D or E.

IWS uses Java regular expression syntax. A tutorial can be found here:
https://docs.oracle.com/javase/tutorial/essential/regex/

REST Wizard (4 of 9)

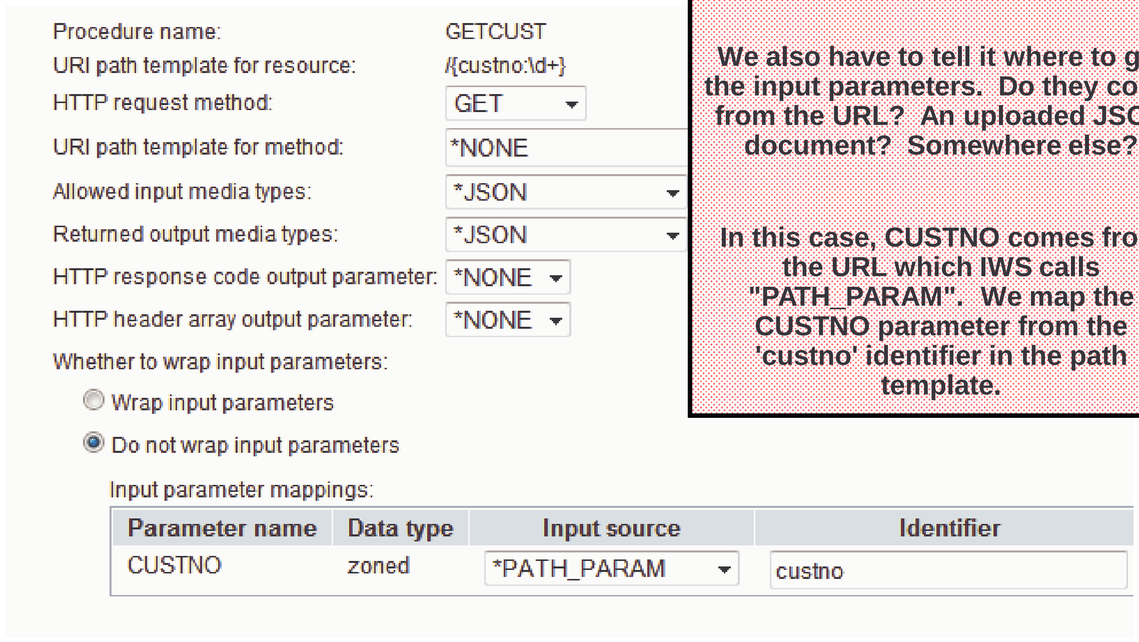


Export procedures:

Select	Procedure name/Parameter name	Usage	Data type
✓	GETCUST		
	CUSTNO	input	zoned
	NAME	output	char
	STREET	output	char
	CITY	output	char
	STATE	output	char
	POSTAL	output	char

Like SOAP, we have to identify which parameters are input or output.

REST Wizard (5 of 9)



Procedure name:	GETCUST
URI path template for resource:	/{custno:\d+}
HTTP request method:	GET
URI path template for method:	*NONE
Allowed input media types:	*JSON
Returned output media types:	*JSON
HTTP response code output parameter:	*NONE
HTTP header array output parameter:	*NONE

Whether to wrap input parameters:

○ Wrap input parameters

● Do not wrap input parameters

Input parameter mappings:

Parameter name	Data type	Input source	Identifier
CUSTNO	zoned	*PATH_PARAM	custno

Here we tell it we want to use GET, and JSON as the data format.

We also have to tell it where to get the input parameters. Do they come from the URL? An uploaded JSON document? Somewhere else?

In this case, CUSTNO comes from the URL which IWS calls "PATH_PARAM". We map the CUSTNO parameter from the 'custno' identifier in the path template.

REST Wizard (steps 6 to 9)

These steps are the same as the SOAP version

STEP 6 = UserID to run the program under

STEP 7 = Library List to run under

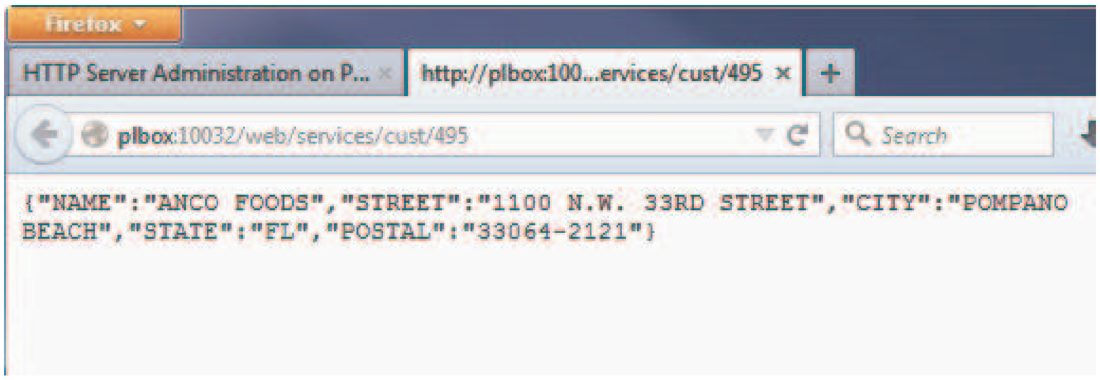
STEP 8 = consumer's IP address or any other HTTP meta data

STEP 9 = Summary screen where you click "Finish" to create the service.

Test REST By Doing a REST Test

When you put a URL into the "location" box in your web browser, the browser does a GET HTTP request. Therefore, a web browser is an easy way to test REST web services that use the GET method.

That way, you can make sure your service works before opening it up to other people who may be using a web service consumer.

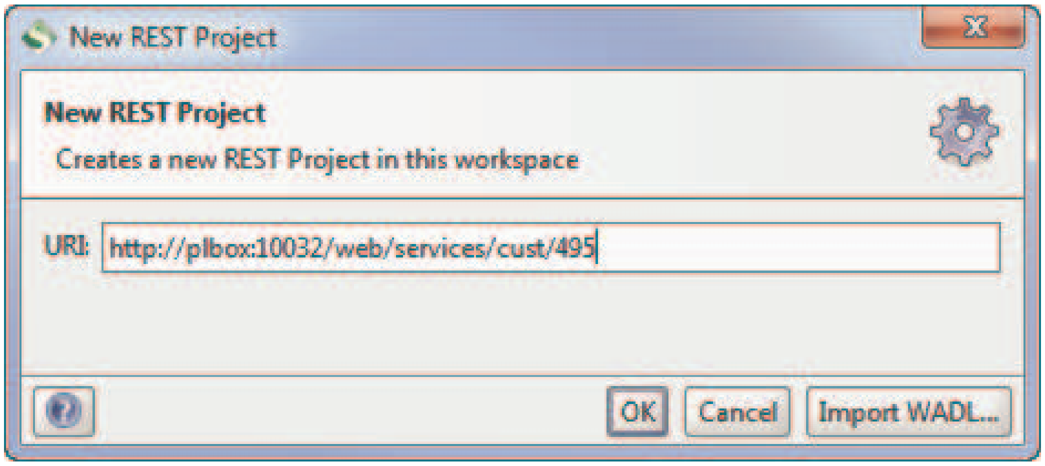

Firefox
HTTP Server Administration on P...
http://plbox:100...ervices/cust/495
plbox:10032/web/services/cust/495
Search

```
{"NAME":"ANCO FOODS","STREET":"1100 N.W. 33RD STREET","CITY":"POMPANO
BEACH","STATE":"FL","POSTAL":"33064-2121"}
```

SOAPUI REST Testing (1 of 2)

Since it's hard to test other methods (besides GET) in a browser, it's good to have other alternatives. Recent versions of SoapUI have nice tools for testing REST services as well.

Choose File / New REST Project, and type the URL, then click OK


New REST Project

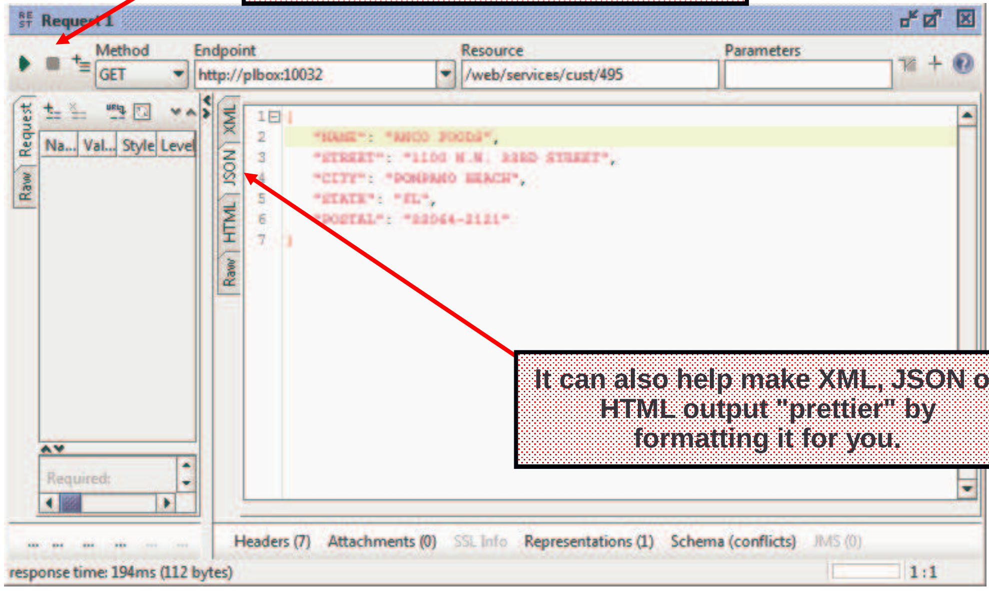
New REST Project

Creates a new REST Project in this workspace

URI: http://plbox:10032/web/services/cust/495

SOAPUI REST Testing (2 of 2)

Here you can change the method and the resource ("noun") easily, and click the green "play" button to try it.


Request 1

Method: GET

Endpoint: http://plbox:10032

Resource: /web/services/cust/495

Parameters:

```
{
   "NAME": "ANCO FOODS",
   "STREET": "1100 N.W. 33RD STREET",
   "CITY": "POMPANO BEACH",
   "STATE": "FL",
   "POSTAL": "33064-2121"
}
```

response time: 194ms (112 bytes)

It can also help make XML, JSON or HTML output "prettier" by formatting it for you.

Do It Yourself

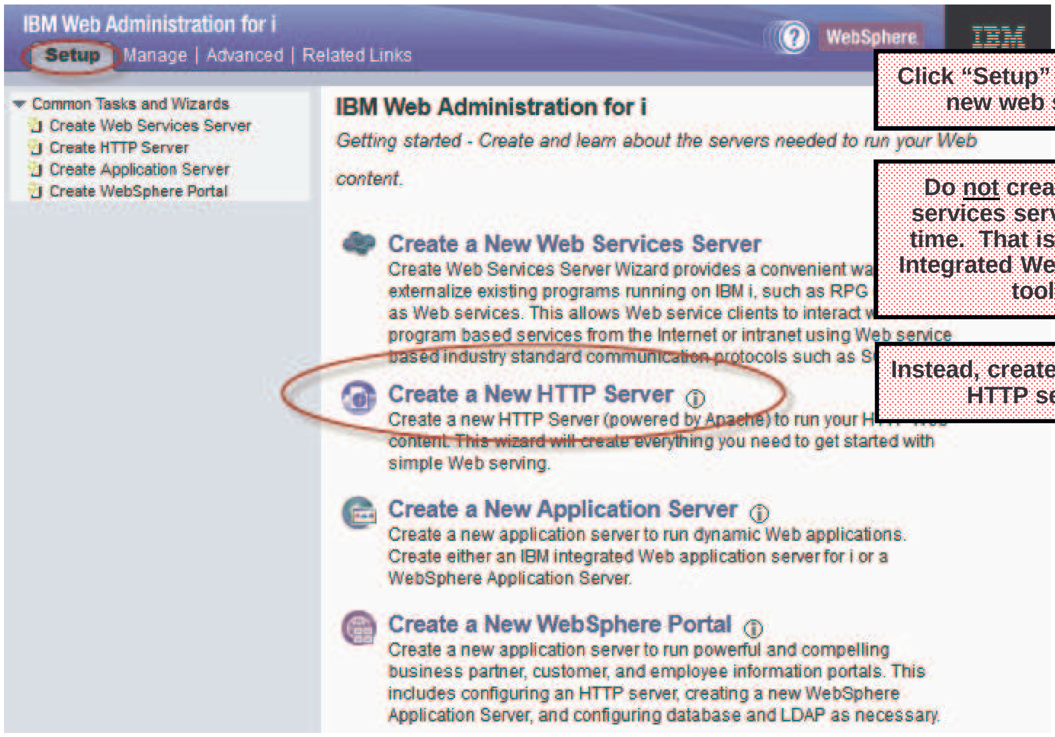
IWS is a neat tool, but:

- Maximum of 7 params
- Can't nest arrays inside arrays
- Supports only XML or JSON
- Very limited options for security
- doesn't always perform well

Writing your own:

- Gives you complete control
- Performs as fast as your RPG code can go.
- Requires more knowledge/work of web service technologies such as XML and JSON
- You can accept/return data in any format you like. (CSV? PDF? Excel? No problem.)
- Write your own security. UserId/Password? Crypto? do whatever you want.
- The only limitation is your imagination.

Create an HTTP Server


IBM Web Administration for i
Setup | Manage | Advanced | Related Links
WebSphere
IBM
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal

IBM Web Administration for i

Getting started - Create and learn about the servers needed to run your Web content.

Create a New Web Services Server
Create Web Services Server Wizard provides a convenient wa externalize existing programs running on IBM i, such as RPG as Web services. This allows Web service clients to interact w program based services from the Internet or intranet using Web service based industry standard communication protocols such as S

Create a New HTTP Server
Create a new HTTP Server (powered by Apache) to run your H content. This wizard will create everything you need to get started with simple Web serving.

Create a New Application Server
Create a new application server to run dynamic Web applications. Create either an IBM integrated Web application server for i or a WebSphere Application Server.

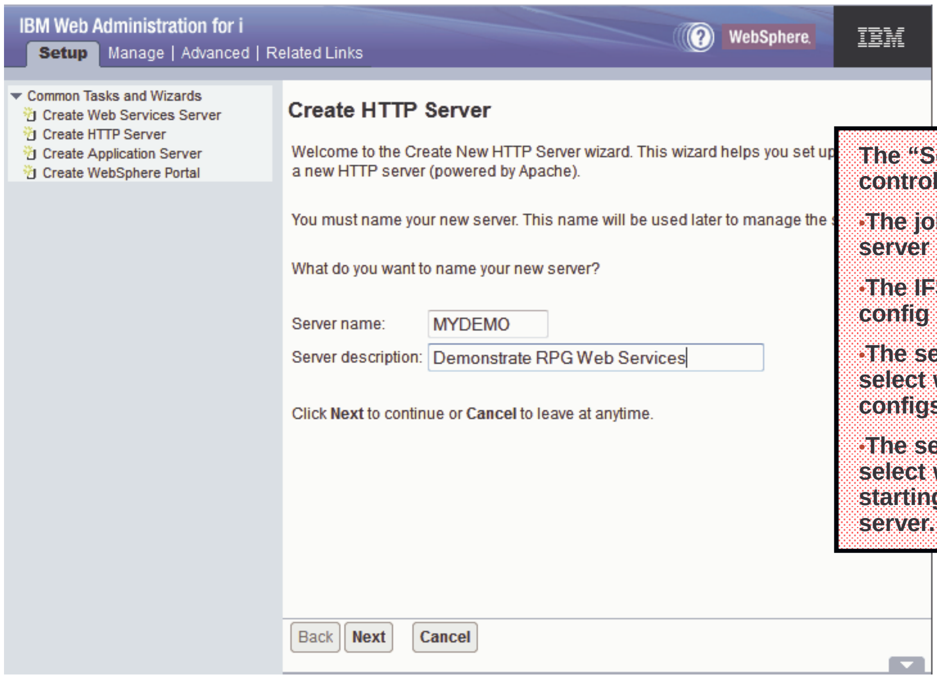
Create a New WebSphere Portal
Create a new application server to run powerful and compelling business partner, customer, and employee information portals. This includes configuring an HTTP server, creating a new WebSphere Application Server, and configuring database and LDAP as necessary.

Click "Setup" to create a new web server.

Do not create a web services server at this time. That is for IBM's Integrated Web Services tool.

Instead, create a "normal" HTTP server.

The “Server Name”


IBM Web Administration for i
Setup | Manage | Advanced | Related Links
WebSphere.
IBM
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Back
Next
Cancel

Create HTTP Server

Welcome to the Create New HTTP Server wizard. This wizard helps you set up a new HTTP server (powered by Apache).

You must name your new server. This name will be used later to manage the s

What do you want to name your new server?

Server name: MYDEMO

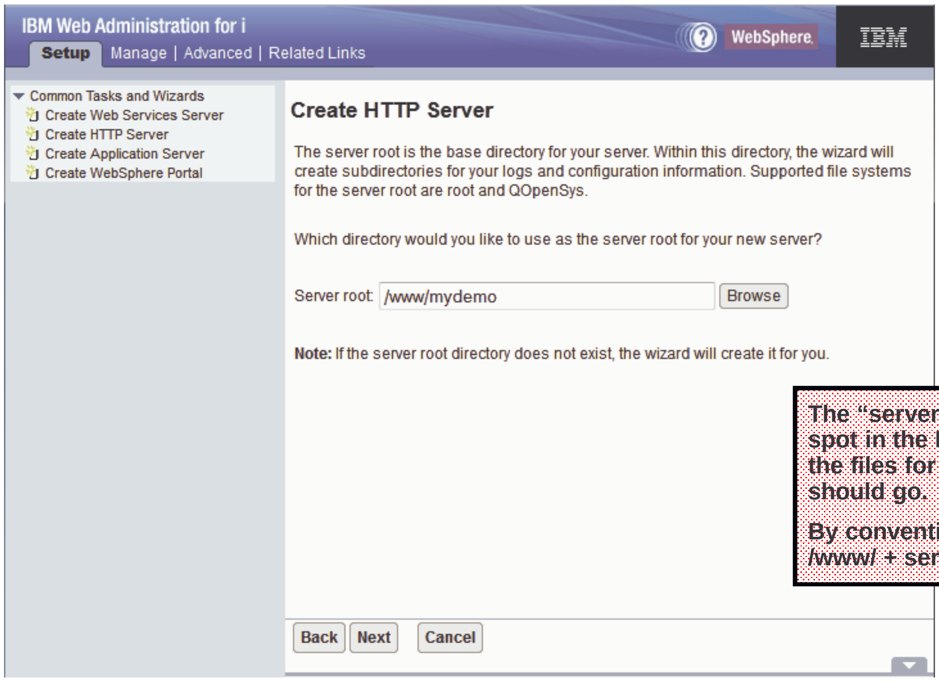
Server description: Demonstrate RPG Web Services

Click **Next** to continue or **Cancel** to leave at anytime.

The “Server Name” controls:

- **The job name of the server jobs**
- **The IFS directory where config is stoed**
- **The server name you select when editing configs**
- **The server name you select when starting/stopping the server.**

Server Root


IBM Web Administration for i
Setup | Manage | Advanced | Related Links
WebSphere.
IBM
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Browse
Back
Next
Cancel

Create HTTP Server

The server root is the base directory for your server. Within this directory, the wizard will create subdirectories for your logs and configuration information. Supported file systems for the server root are root and QOpenSys.

Which directory would you like to use as the server root for your new server?

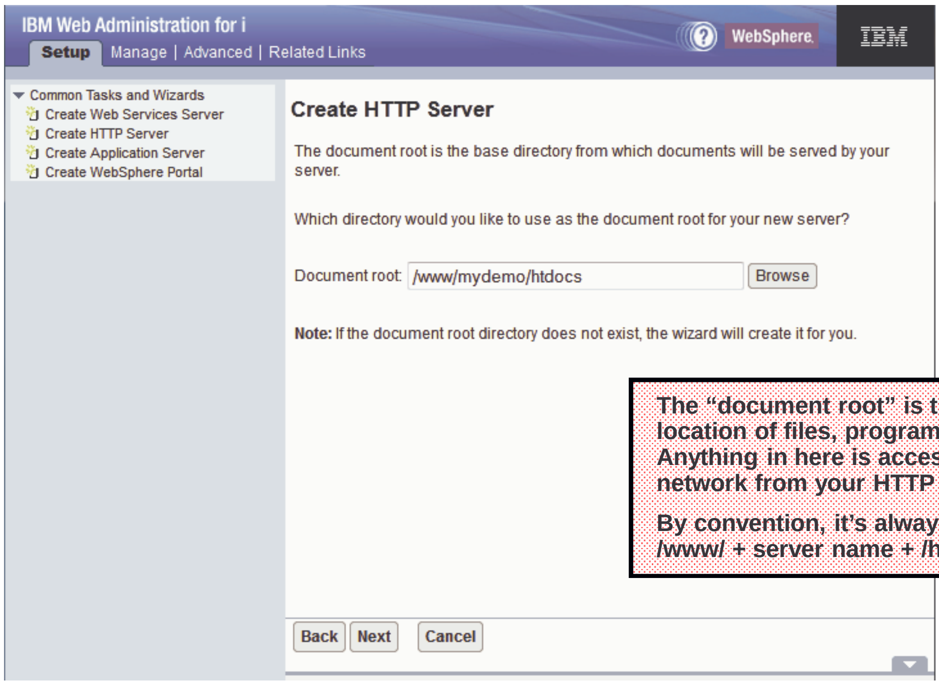
Server root: /www/mydemo

Note: If the server root directory does not exist, the wizard will create it for you.

The “server root” is the spot in the IFS where all the files for this server should go.

By convention, it’s always /www/ + server name.

Document Root


IBM Web Administration for i
Setup | Manage | Advanced | Related Links
WebSphere
IBM
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Browse
Back
Next
Cancel

Create HTTP Server

The document root is the base directory from which documents will be served by your server.

Which directory would you like to use as the document root for your new server?

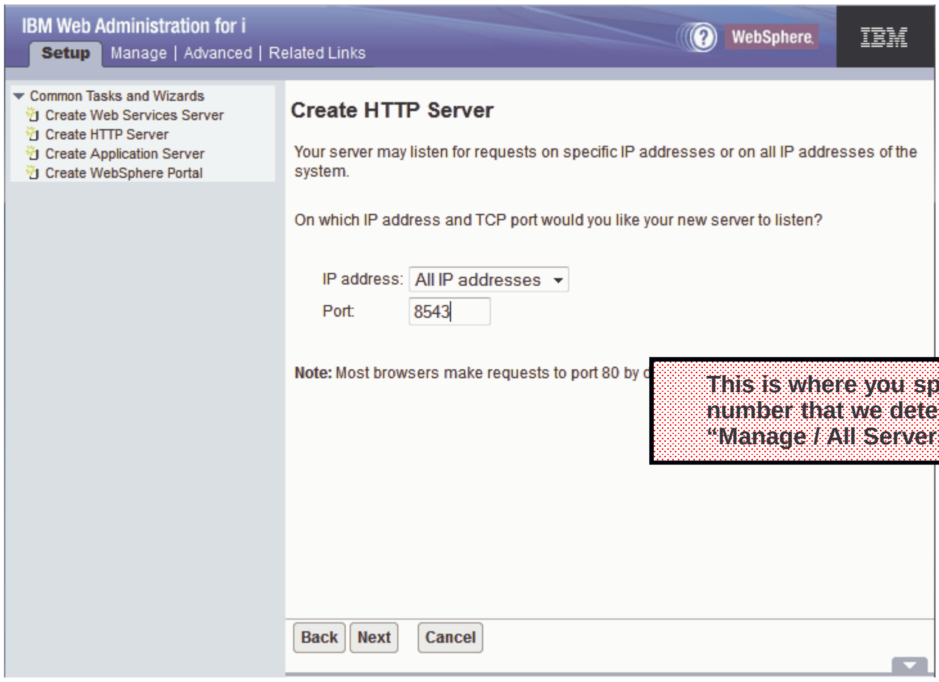
Document root: /www/mydemo/htdocs

Note: If the document root directory does not exist, the wizard will create it for you.

The "document root" is the default location of files, programs, images, etc. Anything in here is accessible over a network from your HTTP server.

By convention, it's always specified as /www/ + server name + /htdocs

Set Port Number


IBM Web Administration for i
Setup | Manage | Advanced | Related Links
WebSphere
IBM
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Back
Next
Cancel

Create HTTP Server

Your server may listen for requests on specific IP addresses or on all IP addresses of the system.

On which IP address and TCP port would you like your new server to listen?

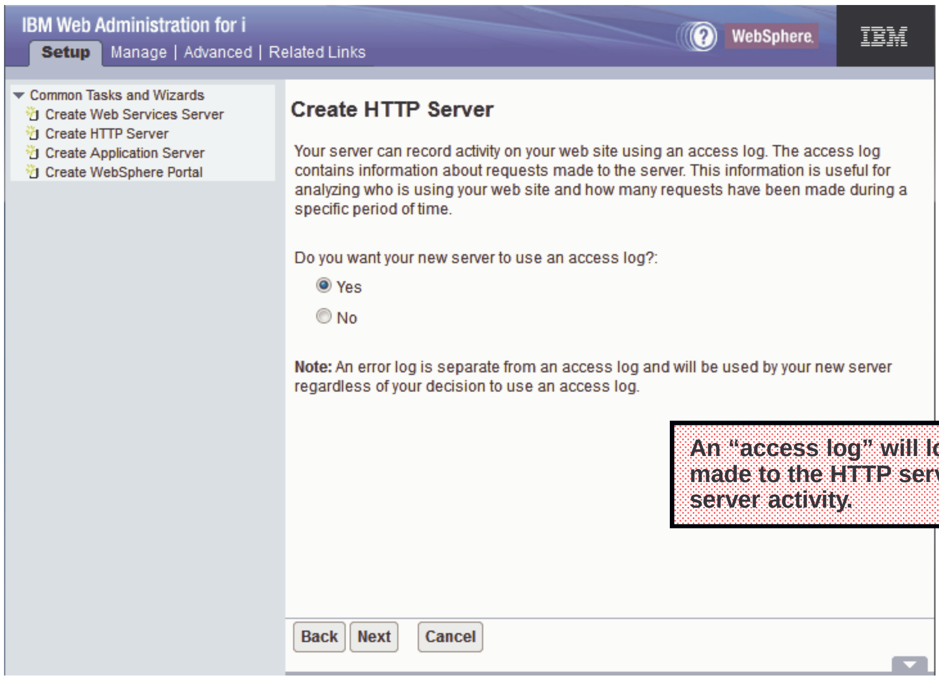
IP address: All IP addresses

Port: 8543

Note: Most browsers make requests to port 80 by d

This is where you specify the port number that we determined on the "Manage / All Servers" screen.

Access Log


IBM Web Administration for i
Setup | Manage | Advanced | Related Links

Common Tasks and Wizards
- Create Web Services Server
- Create HTTP Server
- Create Application Server
- Create WebSphere Portal

Create HTTP Server

Your server can record activity on your web site using an access log. The access log contains information about requests made to the server. This information is useful for analyzing who is using your web site and how many requests have been made during a specific period of time.

Do you want your new server to use an access log?:

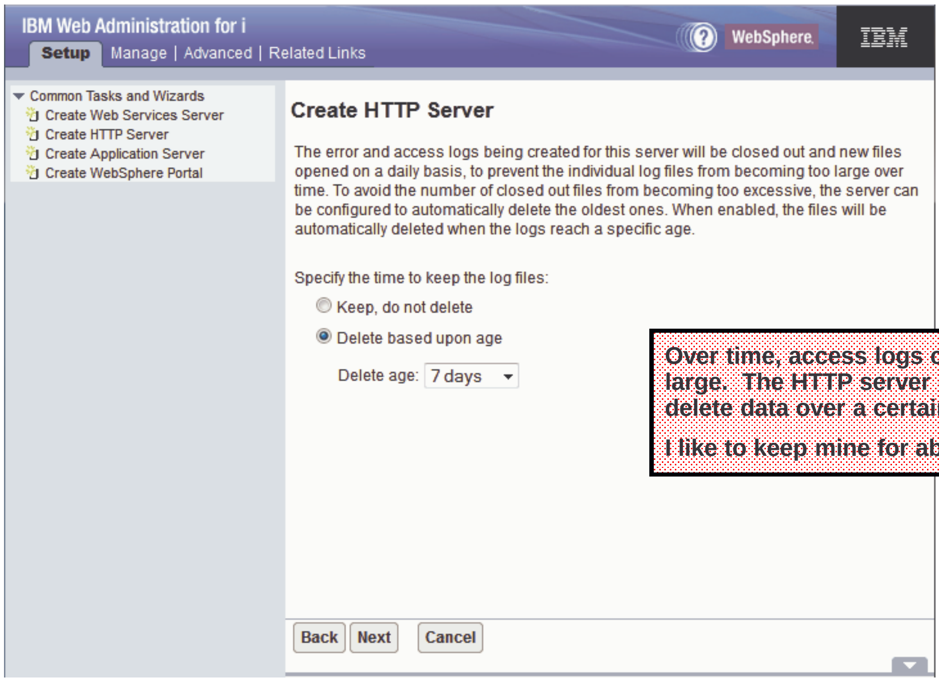
[✓ Yes] No

Note: An error log is separate from an access log and will be used by your new server regardless of your decision to use an access log.

Back | Next | Cancel

An "access log" will log all accesses made to the HTTP server. Useful to track server activity.

Access Log Retension


IBM Web Administration for i
Setup | Manage | Advanced | Related Links

Common Tasks and Wizards
- Create Web Services Server
- Create HTTP Server
- Create Application Server
- Create WebSphere Portal

Create HTTP Server

The error and access logs being created for this server will be closed out and new files opened on a daily basis, to prevent the individual log files from becoming too large over time. To avoid the number of closed out files from becoming too excessive, the server can be configured to automatically delete the oldest ones. When enabled, the files will be automatically deleted when the logs reach a specific age.

Specify the time to keep the log files:

Keep, do not delete [✓ Delete based upon age]

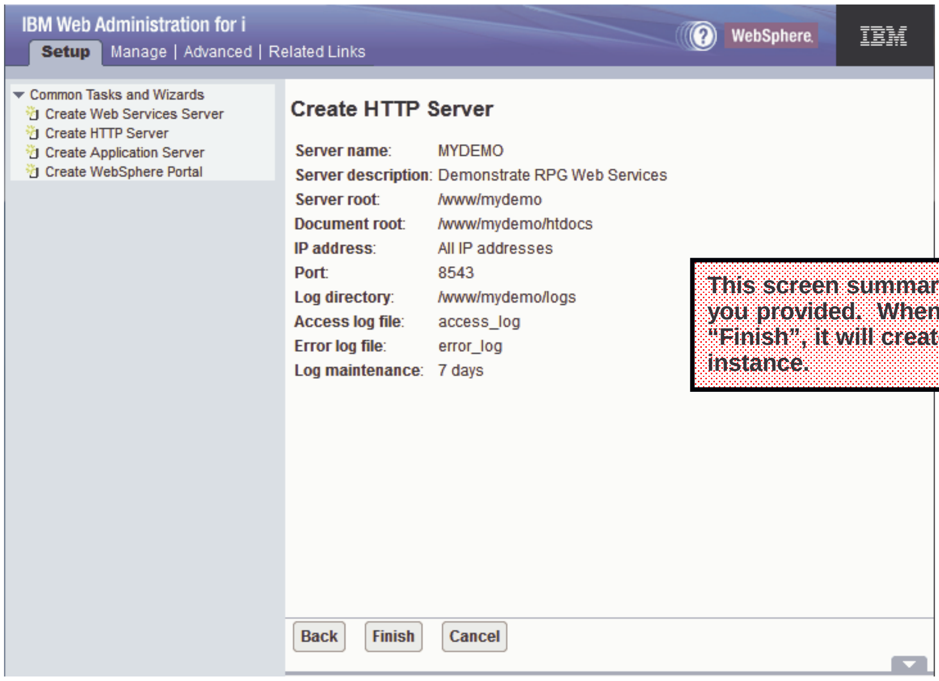
Delete age: 7 days

Back | Next | Cancel

Over time, access logs can get quite large. The HTTP server can automatically delete data over a certain age.

I like to keep mine for about a week.

Summary Screen


IBM Web Administration for i
Setup | Manage | Advanced | Related Links
WebSphere
IBM
Common Tasks and Wizards
Create Web Services Server
Create HTTP Server
Create Application Server
Create WebSphere Portal
Back
Finish
Cancel

Create HTTP Server

Server name:	MYDEMO
Server description:	Demonstrate RPG Web Services
Server root:	/www/mydemo
Document root:	/www/mydemo/htdocs
IP address:	All IP addresses
Port:	8543
Log directory:	/www/mydemo/logs
Access log file:	access_log
Error log file:	error_log
Log maintenance:	7 days

This screen summarizes the settings you provided. When you click "Finish", it will create the server instance.

URL Tells Apache What To Call

To get started with REST, let's tell Apache how to call our program.

You'll want to make a library just for web services, anything in this library will be callable from a web service consumer. I called mine SKWEBSRV

```
ScriptAliasMatch /rest/([a-z0-9]+)/.* /qsys.lib/skwebsrv.lib/$1.pgm

<Directory /qsys.lib/skwebsrv.lib>
   Require all granted
</Directory>
```

- add the preceding code to the bottom of an Apache instance on IBM i.
- **`ScriptAliasMatch`** tells Apache that you want to run a program.
- **`/rest/`** is our ***"context root"*** Apache will **`CALL PGM(SKWEBSRV/XXX)`**
- `[a-z0-9]+` is a regular expression allowing all letters/numbers
- Parenthesis store the value in variable `$1`, used as ***"root resource"***
- The `/.*` allows any trailing characters, we'll use that as our ***"path template"***, and work it out in our RPG program.
- Our REST web service can be run from any IP address (Require all granted).

```
http://your-ibmi:8500/rest/custinfo/495          (CALL SKWEBSRV/CUSTINFO)
```

Apache 2.4 Update

Starting with IBM i 7.2, we have Apache 2.4. They recommend using "require" instead of "Order"

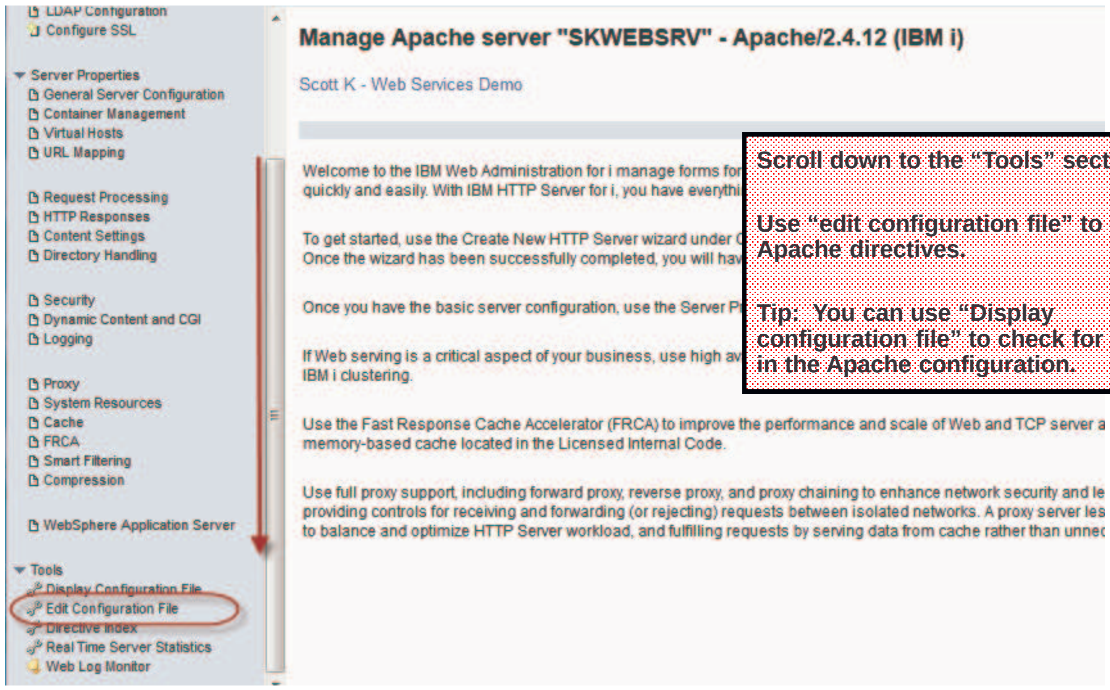
Newer IBM i 7.2 syntax:

```
<Directory /qsys.lib/skwebsrv.lib>
   Require all granted
</Directory>
```

For older releases, replace the above with:

```
<Directory /qsys.lib/skwebsrv.lib>
   Order allow,deny
   Allow from all
</Directory>
```

Edit Configuration File



Manage Apache server "SKWEBSRV" - Apache/2.4.12 (IBM i)

Scott K - Web Services Demo

Welcome to the IBM Web Administration for i manage forms for
quickly and easily. With IBM HTTP Server for i, you have everythi

To get started, use the Create New HTTP Server wizard under C
Once the wizard has been successfully completed, you will hav

Once you have the basic server configuration, use the Server P

If Web serving is a critical aspect of your business, use high av
IBM i clustering.

Use the Fast Response Cache Accelerator (FRCA) to improve the performance and scale of Web and TCP server a
memory-based cache located in the Licensed Internal Code.

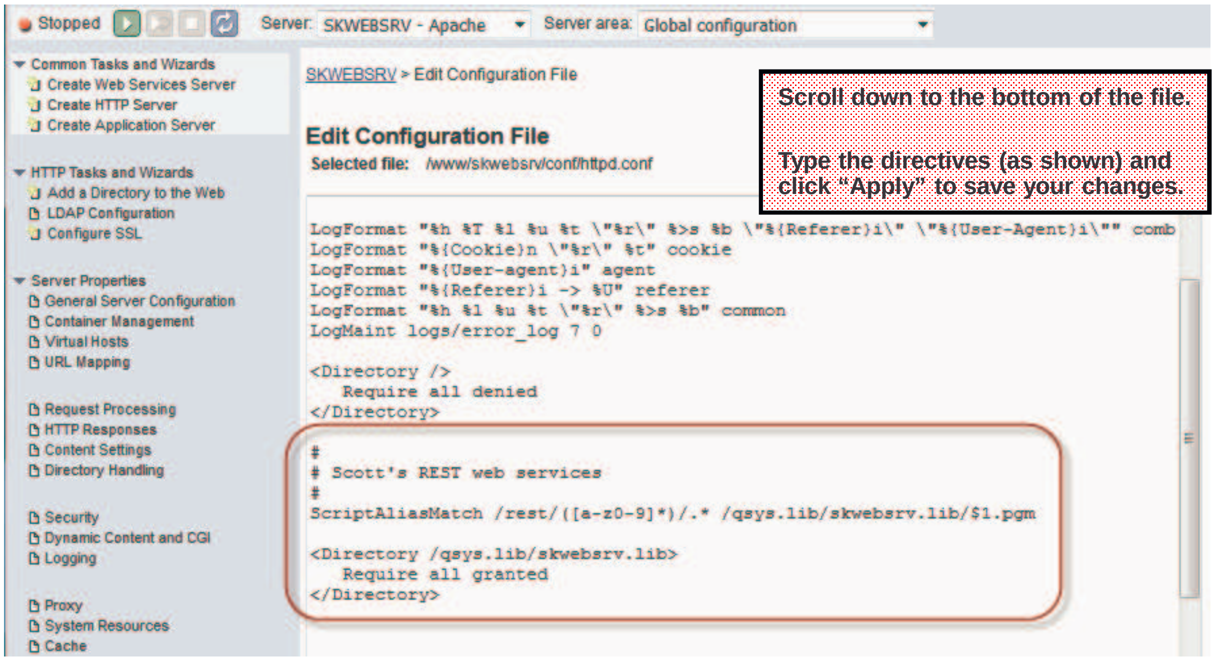
Use full proxy support, including forward proxy, reverse proxy, and proxy chaining to enhance network security and le
providing controls for receiving and forwarding (or rejecting) requests between isolated networks. A proxy server les
to balance and optimize HTTP Server workload, and fulfilling requests by serving data from cache rather than unnec

Scroll down to the "Tools" section.

Use "edit configuration file" to enter Apache directives.

Tip: You can use "Display configuration file" to check for errors in the Apache configuration.

Add Custom Directives



Edit Configuration File

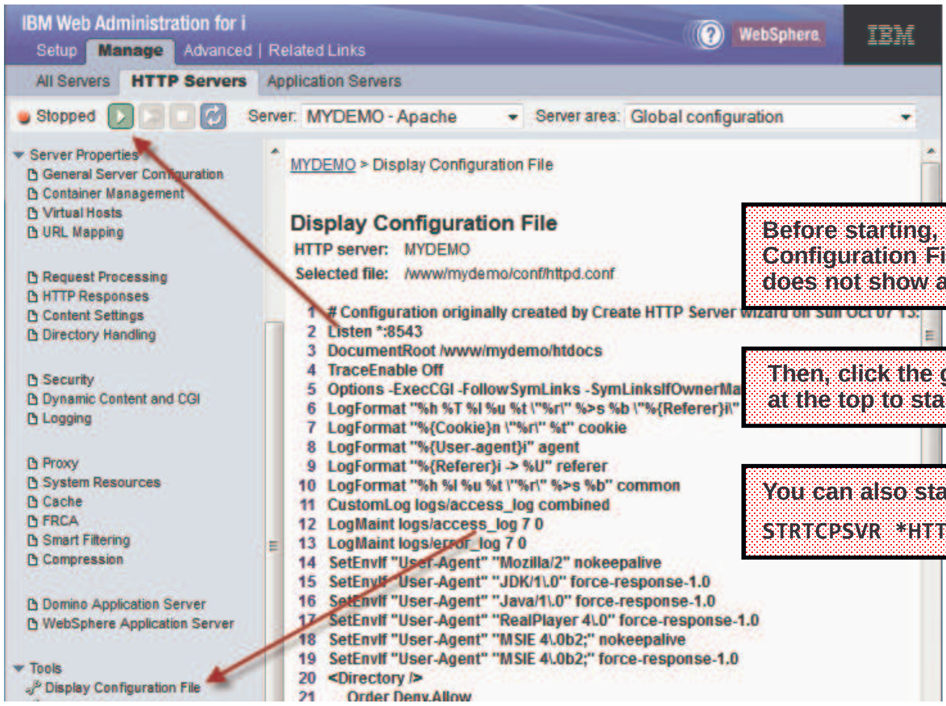
Selected file: /www/skwebsrv/conf/httpd.conf

```
LogFormat "%h %T %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" comb
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogMaint logs/error_log 7 0

<Directory />
   Require all denied
</Directory>

#
# Scott's REST web services
#
ScriptAliasMatch /rest/([a-z0-9]*)/.* /qsys.lib/skwebsrv.lib/$1.pgm

<Directory /qsys.lib/skwebsrv.lib>
   Require all granted
</Directory>
```

Scroll down to the bottom of the file.

Type the directives (as shown) and click "Apply" to save your changes.

Start New Apache Server



Display Configuration File

HTTP server: MYDEMO

Selected file: /www/mydemo/conf/httpd.conf

```
# Configuration originally created by Create HTTP Server wizard on Sun Oct 07 13:
Listen *:8543
DocumentRoot /www/mydemo/htdocs
TraceEnable Off
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMa
LogFormat "%h %T %l %u %t \"%r\" %>s %b \"%{Referer}i\"
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
LogMaint logs/access_log 7 0
LogMaint logs/error_log 7 0
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\.0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\.0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\.0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\.0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\.0b2;" force-response-1.0
<Directory />
    Order Deny,Allow
```

Before starting, click "Display Configuration File" and make sure it does not show any errors.

Then, click the green "start" button at the top to start your new server.

You can also start from 5250 with:

`STRTCPSVR *HTTP HTTPSVR(MYDEMO)`

DIY REST Example

Our web service takes a customer number as input, and returns that customer's address.

Input

```
GET http://your-ibmi:8500/rest/custinfo/495
```

Output

```
<result>
   <cust id="495">
      <name>ANCO FOODS</name>
      <street>1100 N.W. 33RD STREET</street>
      <city>POMPANO BEACH</city>
      <state>FL</state>
      <postal>33064-2121</postal>
   </cust>
</result>
```

This is CGI -- But It's Not HTML

Web servers (HTTP servers) have a standard way of calling a program on the local system. It's know as Common Gateway Interface (CGI)

- The URL you were called from is available via the REQUEST_URI envvar
- The verb GET is available from the REQUEST_METHOD envvar
- If any data is uploaded to your program you can retrieve it from "standard input".
- To write data back from your program to Apache (and ultimately the web service consumer) you write your data to "standard output"

To accomplish this, I'm going to use 3 different APIs (all provided by IBM)
- **`getenv`** ← retrieves an environment variable.
- **`QtmhRdStin`** ← reads standard input
- **`QtmhWrStout`** ← writes data to standard output.

DIY REST Example (1 of 3)

```
FCUSTFILE  IF   E           K DISK

D getenv          PR                 *   extproc('getenv')
D   var                              *   value options(*string)

D QtmhWrStout     PR                     extproc('QtmhWrStout')
D   DtaVar                      65535a   options(*varsize)
D   DtaVarLen                      10I 0 const
D   ErrorCode                    8000A   options(*varsize)

D err             ds                     qualified
D   bytesProv                      10i 0 inz(0)
D   bytesAvail                     10i 0 inz(0)

D xml             pr             5000a   varying
D   inp                          5000a   varying const

D CRLF            C                      x'0d25'
D pos             s                10i 0
D uri             s              5000a   varying
D data            s              5000a
```

DIY REST Example (2 of 3)

```
D ID1             c                   '/custinfo/'

   uri = %str( getenv('REQUEST_URI') );

   monitor;
      pos = %scan(ID1: uri) + %len(ID1);
      custno = %int(%subst(uri:pos));
   on-error;
      data = 'Status: 500 Invalid URI' + CRLF
           + 'Content-type: text/xml' + CRLF
           + CRLF
           + '<error>Invalid URI</error>' + CRLF;
      QtmhWrStout(data: %len(%trimr(data)): err);
      return;
   endmon;

   chain custno CUSTFILE;
   if not %found;
      data = 'Status: 500 Unknown Customer' + CRLF
           + 'Content-type: text/xml' + CRLF
           + CRLF
           + '<error>Unknown Customer Number</error>' + CRLF;
      QtmhWrStout(data: %len(%trimr(data)): err);
      return;
   endif;
```

REQUEST_URI will contain http://x.com/cust/495

Custno is everything after /cust/ in the URL

If an error occurs, I set the status to 500, so the consumer knows there was an error. We also provide a message in XML, in case the consumer wants to show the user.

DIY REST Example (3 of 3)

```
data = 'Status: 200 OK' + CRLF
     + 'Content-type: text/xml' + CRLF
     + CRLF
     + '<result>'
     + '<cust id="' + %char(custno) + '">'
     + '<name>'     + xml(name)     + '</name>'
     + '<street>'   + xml(street)   + '</street>'
     + '<city>'     + xml(city)     + '</city>'
     + '<state>'    + xml(state)    + '</state>'
     + '<postal>'   + xml(postal)    + '</postal>'
     + '</cust>'
     + '</result>'  + CRLF;

QtmhWrStout(data: %len(%trimr(data)): err);
```

Status 200 means that all was well.

Here I send the XML Response.

The xml() subprocedure is just a little tool to escape any special characters that might be in the database fields.

I won't include the code for that in this talk, but you can download the complete program from my web site (see link at end of handout.)

Testing and More Examples

- There's nothing special about testing our DIY example. You call it the same as any other REST web service – just use SoapUI or the browser, just as we did with the IWS example.

- There are additional examples of REST and SOAP in the handout. These are for your benefit – due to time concerns, Scott will skip over these in a standard 75 minute presentation.

REST With Multiple Parameters

- Although the previous slide had only one parameter, REST can have multiple parameters -- but they must all fit on the same URL.

```
http://i.scottklement.com:8001/rest/invoice/495/20100901/20100930
```

- This web service is designed to return a list of invoices for a given customer number, within a given date range.
- 495 = customer number
- 20100901 = start date (in year, month, date format)
- 20100930 = end date (in year, month, date format)

The web service will scan for the slashes, get the parameter info from the URL, and build a JSON document that matches the criteria.

Our JSON Web Service Example

For our next example, we'll create a report of all invoices for a customer.

```
http://i.scottklement.com:8001/rest/invoice/495/20100901/20100930
```

```
{
  "success": false,
  "errmsg": "Put Error Message Here"
}
```

If an error occurs, we'll output a JSON document like this.

```
{ "success": true,
  "errmsg": "",
  "list": [{
     "invno": "xyz",
      "date": "2012-01-23",
      "name": "Acme Industries, Inc.",
    "amount": 123.45,
    "weight": 123.45,
  },
  { same fields again },
  { same fields again },
  { etc }
]}
```

If there's no error, we'll output data in JSON format, as a big array of data structures.

There's no limit to how many rows of data you can place in a JSON document.

DIY JSON, RPG Code (1 of 5)

```
D CRLF            C                   x'0d25'
D data            s           5000a   varying
D uri             s           5000a   varying
D cust            s              4s 0
D sdate           s              8s 0
D edate           s              8s 0
d custpos         s             10i 0
d sdatepos        s             10i 0
d edatepos        s             10i 0
D jsonName        s             25a
D jsonDate        s             10a

 * Unicode versions of {, }, [ and ], respectively.
D LBRACE          C                   u'007b'
D RBRACE          C                   u'007d'
D RSQB            C                   u'005d'
D LSQB            C                   u'005b'

D row             ds                  qualified
D   inv                          5a
D   date                         8s 0
D   name                        25a
D   amount                       9p 2
D   weight                       9p 1
```

DIY JSON, RPG Code (2 of 5)

```
 /free
     exec SQL set option naming=*SYS;

     *inlr = *on;
     uri = %str(getenv('REQUEST_URI'));

     monitor;
        custpos = %scan('/invoice/': uri) + %len('/invoice/');
        sdatepos = %scan('/': uri: custpos) + 1;
        edatepos = %scan('/': uri: sdatepos) + 1;
        cust  = %int(%subst(uri: custpos: (sdatepos-custpos-1)));
        sdate = %int(%subst(uri: sdatepos: (edatepos-sdatepos-1)));
        edate = %int(%subst(uri: edatepos));
     on-error;
        data = 'Status: 500 Invalid URI' + CRLF
             + 'Content-type: text/json' + CRLF
             + CRLF
             + %char(LBRACE) + CRLF
             + '"success": false,' + CRLF
             + '"errmsg": "An unknown URI format was given"' + CRLF
             + %char(RBRACE) + CRLF;
        QtmhWrStout(data: %len(data): err);
        return;
     endmon;
```

DIY JSON, RPG Code (3 of 5)

```
exec SQL declare C1 cursor for
    select aiOrdn, aiIDat, aiSNme, aiDamt, aiLbs
      from ARSHIST
     where aiCust = :cust
       and aiIDat between :sdate
                      and :edate;

exec SQL open C1;
exec SQL fetch next from C1 into :row;

if sqlstt<>'00000'
   and %subst(sqlstt:1:2) <> '01'
   and %subst(sqlstt:1:2) <> '02';
   data = 'Status: 500 Query Failed' + CRLF
        + 'Content-type: text/json' + CRLF
        + CRLF
        + %char(LBRACE) + CRLF
        + '"success": false,' + CRLF
        + '"errmsg": "SQL Failed with SQLSTT='+SQLSTT+'"' + CRLF
        + %char(RBRACE) + CRLF;
   QtmhWrStout(data: %len(data): err);
   return;
endif;
```

DIY JSON, RPG Code (4 of 5)

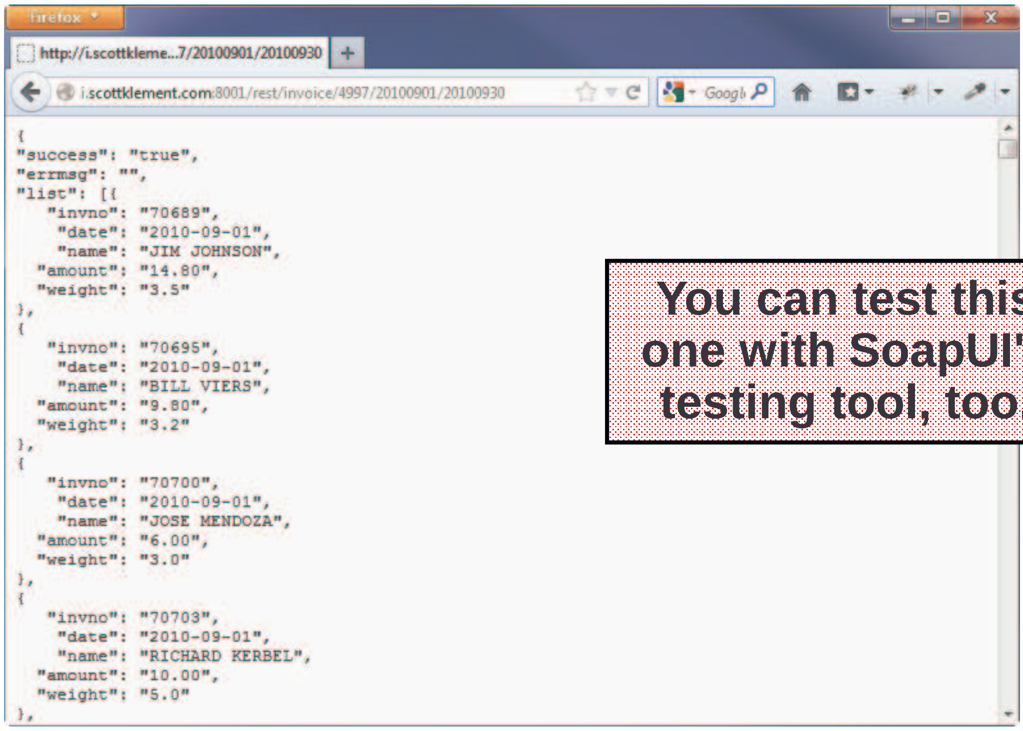
```
data = 'Status: 200 OK' + CRLF
     + 'Content-type: text/json' + CRLF
     + CRLF
     + %char(LBRACE) + CRLF
     + '"success": true,' + CRLF
     + '"errmsg": "",' + CRLF
     + '"list": ' + %char(LSQB);
QtmhWrStout(data: %len(data): err);
```

- Each time I call QtmhWrStout(), it adds more data on to the end of what I've already sent.
- This part is just the start of the JSON document.
- Subsequent calls will write rows of data, and they will be added on to the end.
- Finally, we'll call QtmhWrStout one last time to end the JSON document.

DIY JSON, RPG Code (5 of 5)

```
dow %subst(sqlstt:1:2)='00' or %subst(sqlstt:1:2)='01';
   jsonName = %scanrpl( '"': '\"': row.name );
   jsonDate = %char( %date( row.date: *iso ): *iso );
   data = %char(LBRACE) + CRLF
        + '   "invno": "' + row.inv           + '",' + CRLF
        + '    "date": "' + jsonDate          + '",' + CRLF
        + '    "name": "' + %trim(jsonName)   + '",' + CRLF
        + '  "amount": "' + %char(row.amount) + '",' + CRLF
        + '  "weight": "' + %char(row.weight) + '"'  + CRLF
        + %char(RBRACE);
   QtmhWrStout(data: %len(data): err);

   exec SQL fetch next from C1 into :row;
   if %subst(sqlstt:1:2)='00' or %subst(sqlstt:1:2)='01';
      data = ',' + CRLF;
   else;
      data = CRLF;
   endif;
   QtmhWrStout(data: %len(data): err);
enddo;

data = %char(RSQB) + %char(RBRACE) + CRLF;
QtmhWrStout(data: %len(data): err);
```

JSON Output in Browser


Firefox
http://i.scottkleme...7/20100901/20100930
i.scottklement.com:8001/rest/invoice/4997/20100901/20100930
Google

```
{
"success": "true",
"errmsg": "",
"list": [{
   "invno": "70689",
    "date": "2010-09-01",
    "name": "JIM JOHNSON",
  "amount": "14.80",
  "weight": "3.5"
},
{
   "invno": "70695",
    "date": "2010-09-01",
    "name": "BILL VIERS",
  "amount": "9.80",
  "weight": "3.2"
},
{
   "invno": "70700",
    "date": "2010-09-01",
    "name": "JOSE MENDOZA",
  "amount": "6.00",
  "weight": "3.0"
},
{
   "invno": "70703",
    "date": "2010-09-01",
    "name": "RICHARD KERBEL",
  "amount": "10.00",
  "weight": "5.0"
},
```

You can test this one with SoapUI's testing tool, too.

A SOAP Service With a List

The GETCUST service only returns one "record" so to speak.
Can I do something like the "Invoice List" (the DIY example) using SOAP?

- Q: How do I do that if Idon't code the XML in the program?
- A: With an array!

- Q: How do make an array that returns a list of "records" (more than one field per array element)?
- A: Use an array of data structures.

- Q: What if the number of returned elements (i.e. the number of invoices in the list) varies? How can I specify the number of returned array elements?
- A: If you code a "10i 0" parameter in your parameter list, IWS will let you use it to control the array size.

SOAPINV (invoice list) (1 of 2)

```
H OPTION(*SRCSTMT: *NODEBUGIO) PGMINFO(*PCML:*MODULE)

D row             ds                  qualified inz
D   inv                          5a
D   date                         8s 0
D   name                        25a
D   amount                       9p 2
D   weight                       9p 1

D SOAPINV         PR                  ExtPgm('SOAPINV')
D   CustNo                       4p 0 const
D   strDate                      8p 0 const
D   endDate                      8p 0 const
D   rtnCount                    10i 0
D   rtnList                           likeds(row) dim(999)
D SOAPINV         PI
D   CustNo                       4p 0 const
D   strDate                      8p 0 const
D   endDate                      8p 0 const
D   rtnCount                    10i 0
D   rtnList                           likeds(row) dim(999)
```

This is what needs to be returned for each invoice in the list

rtnCount will tell IWS how many invoices are returned. (to a 999 maximum)

rtnList is the returned array. Notice: LIKEDS!

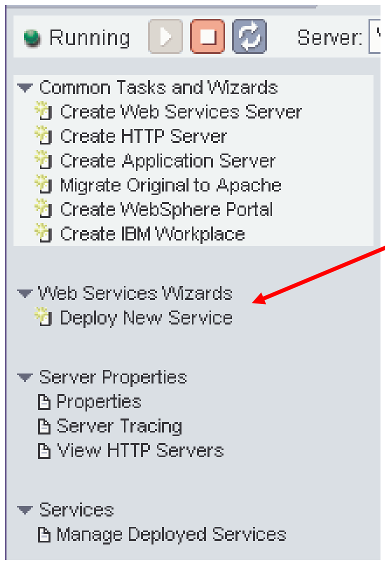
SOAPINV (invoice list) (2 of 2)

```
rtnCount = 0;

exec SQL declare C1 cursor for
    select aiOrdn, aiIDat, aiSNme, aiDamt, aiLbs
      from ARSHIST
     where aiCust = :CustNo
       and aiIDat between :strDate
                      and :endDate;

exec SQL open C1;
exec SQL fetch next from C1 into :row;

dow sqlstt='00000' or %subst(sqlstt:1:2)='01';
   rtnCount = rtnCount + 1;
   rtnList(rtnCount) = row;
   exec SQL fetch next from C1 into :row;
enddo;

exec SQL close C1;
```

CustNo, strDate and endDate are all input parameters passed by IWS.

For each record found, rtnCount is updated, and rtnList() array contains a row.

SOAPINV In the Wizard (1 of 2)

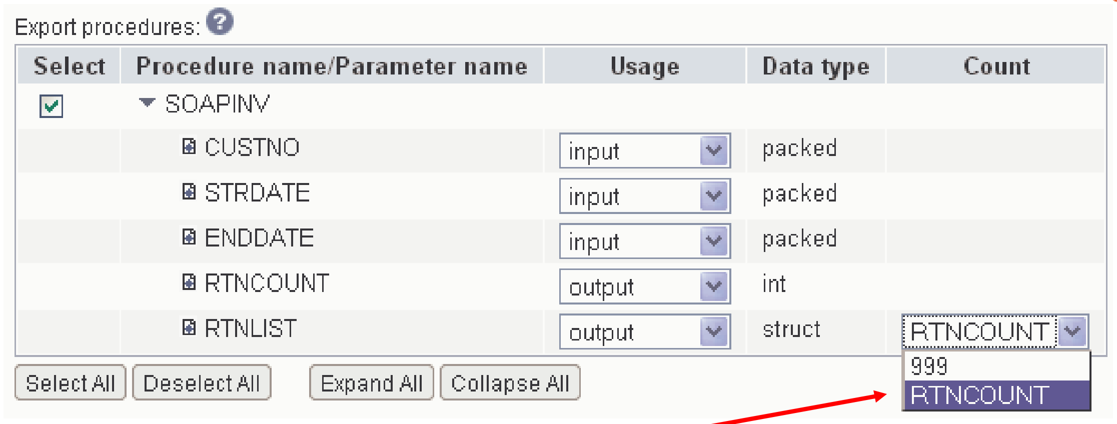


Deploy new service adds another web service to the existing server.

The other screens will be the same as they were for GETCUST.

Except, that on the parameter screen, I have to tell IWS about the returned parameter count.

SOAPINV In the Wizard (2 of 2)



Export procedures:

Select	Procedure name/Parameter name	Usage	Data type	Count
✓	SOAPINV			
	CUSTNO	input	packed	
	STRDATE	input	packed	
	ENDDATE	input	packed	
	RTNCOUNT	output	int	
	RTNLIST	output	struct	RTNCOUNT

By default, the count for RTNLIST is 999, just like the DIM(999) in my RPG code.

But I can change it to "RTNCOUNT" because RTNCOUNT happens to be a 10i 0 field, IWS knows it can be used to specify the array size.

Unfotunately, there's no way to stop IWS from sending RTNCOUNT to the consumer, as well. (But if the consumer doesn't need it, it can ignore it.)

Discussion / Wrap Up

SOAP is heavily standardized, works best with tools
REST is simpler, more versatile, runs faster

- *Use SOAP when making a service to be called my "the masses" (customers, vendors, anything where there are a lot of consumers) because they can use tools, so you don't have to help them.*
- *Use REST for everything else.*
- *Use DIY when you need to go beyond what IWS can do, or when performance is paramount*

More Information / Resources

Gaining a basic understanding of HTTP:

What Is HTTP, Really? (Scott Klement)
http://iprodeveloper.com/application-development/what-http-really

What's the Difference Between a URI, URL, and Domain Name? (Scott Klement)
http://iprodeveloper.com/application-development/whats-difference-between-uri-url-and-domain-name

Gaining a basic understanding of Web Services & Terminology:

Web Services: The Next Big Thing (Scott N. Gerard)
http://iprodeveloper.com/rpg-programming/web-services-next-big-thing

SOAP, WDSL, HTTP, XSD? What? (Aaron Bartell)
http://iprodeveloper.com/rpg-programming/soap-wdsl-http-xsd-what

More Information / Resources

w3schools.com **-- free (and great!) site for learning web technology**

XML:	**http://www.w3schools.com/xml/default.asp**
Web Services:	**http://www.w3schools.com/webservices/default.asp**
WSDL:	**http://www.w3schools.com/wsdl/default.asp**
SOAP:	**http://www.w3schools.com/soap/default.asp**

IBM's web site for the Integrated Web Services (IWS) tool:
http://www.ibm.com/systems/i/software/iws/
http://www.ibm.com/systems/i/software/iws/quickstart_server.html

SoapUI home page
http://www.soapui.org

WSDL2RPG Home Page
http://www.tools400.de/English/Freeware/WSDL2RPG/wsdl2rpg.html

Call a Web Service with WDSL2RPG (Thomas Raddatz)
http://iprodeveloper.com/rpg-programming/call-web-service-wdsl2rpg

More Information / Resources

How-To Articles About Consuming/Providing Web Services:

RPG Consumes the REST (Scott Klement)
http://iprodeveloper.com/rpg-programming/rpg-consumes-rest

RPG Consuming Web Services with HTTPAPI and SoapUI (Scott Klement)
http://iprodeveloper.com/rpg-programming/rpg-consuming-web-services-httpapi-and-soapui

IBM's Integrated Web Services (Scott Klement)
http://iprodeveloper.com/application-development/ibms-integrated-web-services

Consume Web Services with IBM's IWS (Scott Klement)
http://iprodeveloper.com/rpg-programming/consume-web-services-ibms-iws

Serving RESTful Web Services in RPG
http://iprodeveloper.com/rpg-programming/serving-restful-web-services-rpg

Serve JSON Web Services with RPG and YAJL
http://iprodeveloper.com/rpg-programming/serve-json-web-services-rpg-and-yajl

More Information / Resources

Sites that offer web service directories

- WebServiceX.net
- XMethods.net
- BindingPoint.com
- RemoteMethods.com

RPG's XML Opcodes & BIFs:

"Real World" Example of XML-INTO (Scott Klement)
http://iprodeveloper.com/rpg-programming/real-world-example-xml

RPG's XML-SAX Opcode
http://iprodeveloper.com/rpg-programming/rpgs-xml-sax-opcode

PTFs for Version 6.1 Enhance RPG's XML-INTO
http://iprodeveloper.com/rpg-programming/ptfs-version-61-enhance-rpgs-xml

XML-INTO: Maximum Length
http://iprodeveloper.com/rpg-programming/xml-maximum-length

XML-INTO: Read XML Data Larger Than 65535
http://iprodeveloper.com/rpg-programming/xml-read-xml-data-larger-65535

XML-INTO: Output to Array Larger than 16 MB
http://iprodeveloper.com/rpg-programming/xml-output-array-larger-16-mb

This Presentation

You can download a PDF copy of this presentation from:

http://www.scottklement.com/presentations/

The Sample Web Service Providers in this article are also available at the preceding link.

Thank you!