

# Providing RPG Web Services



## on IBM i

Presented by

Scott Klement

<http://www.scottklement.com>

© 2012-2025, Scott Klement

*"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:



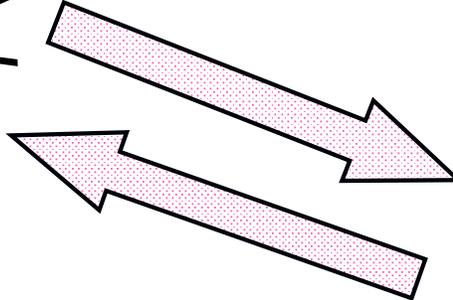
1. Introduction
  - How do they work?
  - What are JSON and XML?
2. REST web service with IBM's IWS
3. Writing your own from the ground-up with Apache.
4. Discussion/wrap-up

# How Do They Work?



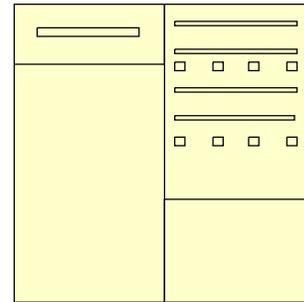
HTTP starts with a request for the server

- 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.



# JSON and XML to Represent a DS



```
D list          ds          qualified
D              ds          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 Supply 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!

9

## IBM's Integrated Web Services Server



IBM provides a Web Services (aka Web API, aka REST API) tool with IBM i at no extra charge!

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

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

<https://www.ibm.com/support/pages/integrated-web-services-ibm-i-web-services-made-easy>

Requirements:

- IBM i operating system
- 57xx-SS1, opt 30: QShell
- 57xx-SS1, opt 33: PASE
- 57xx-JV1, opt 14 (or higher): Java
- 57xx-DG1 -- the HTTP server (powered by Apache)

*Make sure you have the latest TR, cum & group PTFs installed.*

10

# Let's Get Started!



## The HTTP server administration tool runs in IBM Navigator for i

- 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" (old nav)  
or "Bookmarks" (new nav)
- Click "IBM Web Administration for i"

11

## IBM Navigator for i (old nav)



Firefox

plbox - IBM Navigator for i

https://plbox:2005/ib

Target  
Welcome sklement system: Help | Logout

IBM® Navigator for i

Welcome

Search Task

IBM i Management

- Target Systems and Groups
- Favorites
- System
- Monitors
- Basic Operations
- Work Management
- Configuration and Services
- Network
- Integrated Server Administration
- Security
- Users and Groups
- Database
- Journal Management
- Performance
- File Systems
- Internet Configurations

Settings

Welcome

Welcome to the IBM Navigator for i [About Console](#)

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.

[IBM i Tasks Page](#)

Click "Internet Configurations"

12

# Internet Configurations (old nav)



The screenshot shows a Firefox browser window displaying the IBM Navigator for i interface. The address bar shows the URL <https://plbox.2005/ib>. The page title is "Internet Configurations - Plbox". The main content area lists several configuration options:

- [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.

A red arrow points from a box labeled "IBM Web Administration for i" to the corresponding link in the list. A "Close" button is visible at the bottom of the configuration list.

13

# IBM Navigator for i (new nav)



The screenshot shows the IBM Navigator for i dashboard. The title bar reads "IBM Navigator for i". On the left side, there is a vertical toolbar with icons for various functions. The main content area displays a summary card for the system:

**i - Active**

CPU:	0.00
Active jobs:	276
System ASP %:	37.02 <span style="color: green;">✔</span>

A red arrow points from a box labeled "Double-Click the IBM i system to work with" to the "i - Active" card.

14

# Bookmarks (new nav)



Open the "Bookmarks" item in the lower-left, and click "IBM Web Administration for i"

IBM Web Administration for i

CPU Utilization (%)

0.5 -  
0.4 -  
0.3 -  
0.2 -  
0.1 -

15

# Web Administration for i



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

**IBM Web Administration for i**

Getting started - Create and learn about the servers needed

**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.

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)



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 Web Services Server

Specify Web services server name - Step 1 of 4

Welcome to the Create Web Services Server wizard. A 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 via Web service based industry standard communication protocols such as SOAP and REST. 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-01.ibm.com/support/docview.wss?uid=isg3T1026868>

Specify a unique name for this server ?

Server name: SKWEBSERV

Server description: Scott K's Web Services

Create HTTP server

Back Next Cancel

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

**Description** can be whatever you want... should explain what the server is to be used for.

# Create IWS Server (2 of 4)



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 Web Services Server

Specify network attributes for server - Step 2 of 4

Your server may listen for requests on specific IP addresses or on all IP addresses of the system. A command port is used to manage the server.

Specify internet addresses and ports for server ?

Specify server command port: 10107

Specify internet address and port for the server

IP address: All IP addresses

Port: 10106

Specify internet address and port for the HTTP server

IP address: All IP addresses

Port: 10116

Back Next Cancel

**Two servers are needed**

1. One to run Java (application server)
2. One that handles the web communications (HTTP server)

A third port is used to communicate commands between them.

Port numbers must be unique system-wide.

The wizard will provide defaults that should work.

# Create IWS Server (3 of 4)



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 Web Services Server

Specify User ID for Server - Step 3 of 4

The server requires an IBM i user ID to run the server's jobs. It is recommended that a special user ID is specified to run the server's jobs since this user ID is given authority to all of the 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 QWSERVICE.

Specify an existing user ID

Create a new user ID

Back Next Cancel

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)



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 Web Services Server

Summary - Step 4 of 4

Servers Service

Web Services Server Information

Server name: SKWEBSERV  
Server description: Scott K's Web Services  
Port: 10106  
Command port: 10107  
Server root: /www/SKWEBSERV  
Server URL: http://power8.profoundnet.local:10116  
User ID for server: QWSERVICE  
Context root: /web

HTTP Server Information

Back Finish Cancel

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!



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.

# GETCUST RPG Program (1 of 2)



```
Ctl-Opt DFTACTGRP(*NO) ACTGRP('WEBAPI') PGMINFO(*PCML:*MODULE);
Dcl-F CUSTFILE Usage(*Input) Keyed PREFIX('CUST. ');
Dcl-DS CUST ext extname('CUSTFILE') qualified End-DS;

Dcl-PI *N;
  CustNo           like(Cust.Custno);
  Name             like(Cust.Name);
  Street           like(Cust.Street);
  City             like(Cust.City);
  State            like(Cust.State);
  Postal           like(Cust.Postal);
End-PI;

Dcl-PR QMHSNDPM ExtPgm('QMHSNDPM');
  MessageID       Char(7)   Const;
  QualMsgF        Char(20)  Const;
  MsgData         Char(32767) Const options(*varsize);
  MsgDtaLen       Int(10)   Const;
  MsgType         Char(10)  Const;
  CallStkEnt      Char(10)  Const;
  CallStkCnt      Int(10)   Const;
  MessageKey      Char(4);
  ErrorCode       Char(8192) options(*varsize);
End-PR;
```

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.

Since there's no DCL-PROC, the DCL-PI works like the old \*ENTRY PLIST

## GETCUST RPG Program (2 of 2)



```
Dcl-DS err qualified;
  bytesProv      Int(10)   inz(0);
  bytesAvail     Int(10)   inz(0);
End-DS;

Dcl-S MsgDta      Varchar(1000);
Dcl-S MsgKey      Char(4);
Dcl-S x           Int(10);

chain CustNo CUSTFILE;
if not %found;
  msgdta = 'Customer not found.';
  QMHSNDPM( 'CPF9897': 'QCPFMSG *LIBL': msgdta:
%len(msgdta): '*ESCAPE'
           : '*PGBDY': 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;
```

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 creating JSON or XML for me!

29

## 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 an XML document that is generated by the RPG compiler.

### *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)
          INFOTMF('/path/to/myfile.pcml')
```

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

```
Ctl-Opt PGMINFO(*PCML:*MODULE);
```

30

# GETCUST as a REST API



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 **C**reate, **R**ead, **U**ppdate, **D**ele...)
- 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.

31

# 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.

32

# Deploy a New REST API



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

All Servers | HTTP Servers **Application Servers** Installations

Running [Stop] [Refresh] Server: SKIWS1 - V2.6 (web services) ▾

SKIWS1 > Manage Deployed Services

**Manage Deployed Services**

Data current as of Apr 20, 2023 5:37:08 AM.

Deployed services: ?

	Service name	Status	Type	Startup type	Service definition
<input type="radio"/>	ConvertTemp	<span style="color: green;">●</span> Running	SOAP	Automatic	View WSDL

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

33

# REST Wizard (1 of 10)



The type (dropdown) should be REST.

You can use a program or SQL statement – for this example, I'll specify an ILE program and type the IFS path of the GETCUST program.

Running [Stop] [Refresh] Server: SKIWS1 - V2.6 (web services) ▾

SKIWS1 > Manage Deployed Services > Deploy New Service

**Deploy New Service**

Specify Web service type - Step 1 of 10

Welcome to the Deploy New Service wizard. This wizard helps you create Web services using IBM i objects, messages that are based on the SOAP protocol. A REST-based Web service exposes resources, where

Specify Web service type: REST ▾ ?

Specify Web service implementation:

ILE program object as a Web service

Specify path to ILE program or service program: ?

Path of program object: /QSYS.LIB/SKWEBSRV.lib/GETCUST.pgm  e.g. /Q

**Note:** Specify a \*PGM or \*SRVPGM object.

SQL as a Web service

34

## REST Wizard (2 of 10)



Running Server: SKIWS1 - V2.6 (web services) ▾

Common Tasks and Wizards

- Web Services
  - Deploy New Service
  - Manage Deployed Services
- Server Properties
  - Properties
  - View HTTP Servers
- Security
- Logging
  - View Logs
  - View Create Summary
- Tools
  - Web Log Monitor
  - Create Certificate
  - Manage Certificates
  - Create Keystore

SKIWS1 > Manage Deployed Services > Deploy New Service

### Deploy New Service

Specify Name for Service - Step 2 of 10

The Web service to be externalized is a resource.

Resource name:

Service description:

URI path template:

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

description can be  
whatever you want.

PATH template deserves  
its own slide 😊

35

## 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

36

## REST Wizard (3 of 10)



Running Server: SKIWS1 - V2.6 (web services) ▼

Common Tasks and Wizards

- Web Services
  - Deploy New Service
  - Manage Deployed Services
- Server Properties
  - Properties
  - View HTTP Servers
- Security
- Logging
  - View Logs
  - View Create Summary
- Tools
  - Web Log Monitor
  - Create Certificate
  - Manage Certificates
  - Create Keystore

SKIWS1 > Manage Deployed Services > Deploy New Service

### Deploy New Service

Specify security constraint - Step 3 of 10

The security constraint limits who can access the service.

Secure transport required: No ▼

Protect using authentication method: \*NONE ▼

\*NONE

\*BASIC

Secure transport determines whether or not SSL (TLS) is required.

Authentication method \*BASIC will require a userid/password.

37

## 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/>

38

# REST Wizard (4 of 10)



## Deploy New Service

Select Export Procedures to Externalize as a Web Service - Step 4 of 10

Exported procedures are entry points to a program object and are mapped to Web service operations. A procedure is a set of only one procedure.

The table below lists all the exported procedures found in the program object that can be externalized through this Web service the Web service.

Detect length fields

Use parameter name as element name for data structures

Export procedures: ?

Select	Procedure name/Parameter name	Usage	Data type
<input checked="" type="checkbox"/>	▼ GETCUST		
	CUSTNO	input ▼	zoned
	NAME	output ▼	char
	STREET	output ▼	char
	CITY	output ▼	char
	STATE	output ▼	char
	POSTAL	output ▼	char

Select All Deselect All Expand All Collapse All

"Detect length fields" will look for fields named ending with `_LENGTH` and treat them as the number of elements for any arrays.

We also need to tell it which parameters are used for input and output from our program.

39

# REST Wizard (5 of 10)



Running Server: SKIWS1 - V2.6 (web services) ▼

Common Tasks and Wizards

- Web Services
  - Deploy New Service
  - Manage Deployed Services
- Server Properties
  - Properties
  - View HTTP Servers
- Security
- Logging
  - View Logs
  - View Create Summary
- Tools
  - Web Log Monitor
  - Create Certificate
  - Manage Certificates
  - Create Keystore

SKIWS1 > Manage Deployed Services > Deploy New Service

### Deploy New Service

Specify ILE Procedure Information - Step 5 of 10

Customize how each procedure invocation handles web service calls. ?

Procedure name: GETCUST

Trim mode for character fields: Trailing ▼

User-defined error message:

HTTP status code on procedure call success: 200 or... ▼

HTTP status code on procedure call failure: 500 or... ▼

We can control how blanks are trimmed from character fields.

We can also control which HTTP status codes are returned for success/failures.

40

## REST Wizard (6 of 10)



Security

- Logging
- View Logs
- View Create Summary

Tools

- Web Log Monitor
- Create Certificate
- Manage Certificates
- Create Keystore

Procedure name: GETCUST  
URI path template for resource: /{custno:id+}  
HTTP request method: GET  
URI path template for method: \*NONE  
HTTP response code output parameter: \*NONE  
HTTP header array output parameter: \*NONE  
HTTP header information: \*NONE

Error response output parameter: \*NONE or...  
Allowed input media types: \*ALL or...  
Returned output media types: \*JSON or...  
Identifier for input wrapper element: GETCUSTInput or...  
Identifier for output wrapper element: GETCUSTResult or...

Wrap output parameters  
 Wrap input parameters

Input parameter mappings:

Parameter name	Data type	Input source	Identifier	Default Value
CUSTNO	zoned	*PATH_PARAM	custno	*NONE

Back Next Cancel

Since this example just retrieves a customer, I used the "GET" method.

The output document will be JSON.

The input parameter comes from the "Path" portion of the URL.

## REST Wizard (7 of 10)



SKIWS1 > Manage Deployed Services > Deploy New Service

### Deploy New Service

Specify User ID for this Service - Step 7 of 10

The service requires an IBM i user ID to run the Web service business logic. The user ID must have the necessary au

Specify User ID for this Service: ?

- Use server's user ID
- Specify an existing user ID
- Use authenticated user ID

Similar to when the server was created, we can specify which userid this particular API will run under.

The most secure method is to create a user specially for this, and give it the minimum possible authority for the API to work.

# REST Wizard (8 of 10)



## Deploy New Service

Specify Library List - Step 8 of 10

The functionality of the IBM i program you want to externalize as a Web service may depend upon other IBM i progra

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
<input type="radio"/>	SKWEBSRV

Add Remove All

This step lets you configure a library list that will be in effect when the API is run.

# REST Wizard (9 of 10)



## Deploy New Service

Specify Transport Information to Be Passed - Step 9 of 10

Specify transport information to be passed to the web service implementation code. ?

Specify Transport Metadata:

	Transport Metadata
<input type="checkbox"/>	QUERY_STRING
<input type="checkbox"/>	REMOTE_ADDR
<input type="checkbox"/>	REMOTE_USER
<input type="checkbox"/>	REQUEST_METHOD
<input type="checkbox"/>	REQUEST_URI
<input type="checkbox"/>	REQUEST_URL
<input type="checkbox"/>	SERVER_NAME
<input type="checkbox"/>	SERVER_PORT

This screen lets you control which environment variables will be set when the API runs.

This is a bit more "advanced", but if you wanted to know the IP address of the API consumer, for example, you could enable the REMOTE\_ADDR variable, then retrieve that variable in your RPG program.

Specify HTTP Headers:

	HTTP Headers
There are no entries for this table.	

Add Remove All

# REST Wizard (10 of 10)



## Deploy New Service

Summary - Step 10 of 10

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

Service Security Methods Request Information

Resource name: cust  
Resource description: Retrieve Customer  
Service install path : /www/skiws1/webservices/services/cust  
URI path template: /{custno:\d+}  
Program: /QSYS.LIB/SKWEBSRV.LIB/GETCUST.PGM  
Library list for service: SKWEBSRV

The last step shows all of the options you selected (for your review).  
When you click **FINISH** it will create the REST API

Back Finish Cancel

45

# Wait For the API to Install



IBM Web Administration for i

Setup **Manage** Advanced | Related Links

All Servers | HTTP Servers **Application Servers** Installations

Running Server: SKIWS1 - V2.6 (web services)

Common Tasks and Wizards

- Web Services
  - Deploy New Service
  - Manage Deployed Services
- Server Properties
  - Properties
  - View HTTP Servers
- Security
- Logging
  - View Logs
  - View Create Summary
- Tools
  - Web Log Monitor
  - Create Certificate
  - Manage Certificates
  - Create Keystore

SKIWS1 > Manage Deployed Services

### Manage Deployed Services

Data current as of Apr 20, 2023 6:05:32 AM.

Deployed services:

	Service name	Status	Type	Startup type	Service definition
<input type="radio"/>	ConvertTemp	Running	SOAP	Automatic	<a href="#">View WSDL</a>
<input checked="" type="radio"/>	cust	Installing	REST	Automatic	<a href="#">View Swagger</a>

Deploy Properties Uninstall Redeploy Refresh

The hourglass indicates that creating the API is in progress. Click "Refresh" a couple of times until it shows as "Running"

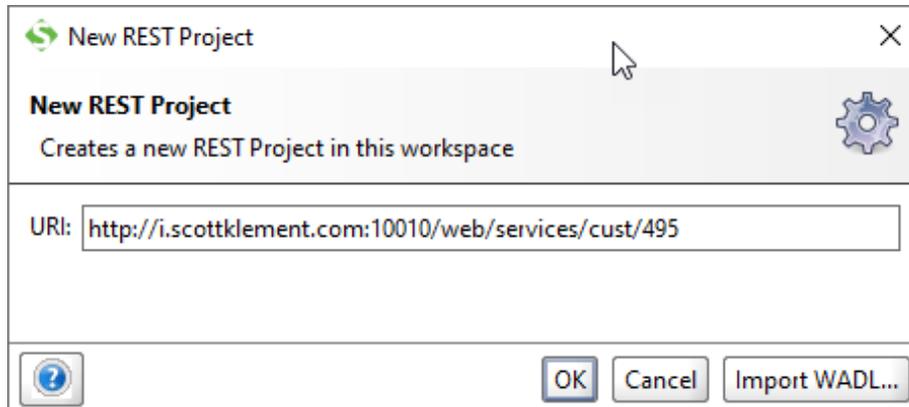
46

## 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



47

## 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.

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

response time: 190ms (108 bytes)

48

# Do It Yourself



IWS is a neat tool, but:

- 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



The screenshot shows the IBM Web Administration for i interface. The top navigation bar includes 'Setup', 'Manage', 'Advanced', and 'Related Links'. A sidebar on the left lists 'Common Tasks and Wizards' with options: 'Create Web Services Server', 'Create HTTP Server', and 'Create Application Server'. The main content area is titled 'IBM Web Administration for i' and 'Getting started - Create and learn about the servers needed to...'. It lists three options:

- Create a New Web Services Server**: Create Web Services Server Wizard provides a convenient way to create a new Web Services Server. This allows Web service clients to interact with IBM i programs from the Internet or intranet using Web service based industry 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 on an IBM integrated Web application server for i or a WebSphere...

Callouts (text boxes) are overlaid on the screenshot:

- Top right: Click "Setup" to create a new web server.
- Middle right: Do **not** create a web services server at this time. That is for IBM's Integrated Web Services tool, currently used only for SOAP.
- Bottom right: Instead, create a "normal" HTTP server.
- A red circle highlights the 'Create a New HTTP Server' option.

# The “Server Name”



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

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 server.

What do you want to name your new server?

Server name:

Server description:

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 stored
- 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

Common Tasks and Wizards

- Create Web Services Server
- Create HTTP Server
- Create Application Server
- Create WebSphere Portal

### 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:

**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

Common Tasks and Wizards

- Create Web Services Server
- Create HTTP Server
- Create Application Server
- Create WebSphere Portal

### 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:

**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

Common Tasks and Wizards

- Create Web Services Server
- Create HTTP Server
- Create Application Server
- Create WebSphere Portal

### 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:

Port:

**Note:** Most browsers make requests to port 80 by default.

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

WebSphere IBM

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.

55

# Access Log Retention



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

### 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.

56

# 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

### 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

Back Finish Cancel

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

57

# URL Tells Apache What to Call



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

```
ScriptAlias /cust /qsys.lib/restful.lib/custinfo.pgm
<Directory /qsys.lib/restful.lib>
  Require all granted
</Directory>
```

- Just add the preceding code to an already working Apache instance on IBM i.
- **ScriptAlias** tells apache that you want to run a program.
- If URL starts with /cust, Apache will **CALL PGM(RESTFUL/CUSTINFO)**
- Our REST web service can be run from any IP address (Allow from all).

```
http://ibmi.example.com/cust/495
```

- Browser connects to: **ibmi.example.com**
- Apache sees the /cust and calls RESTFUL/CUSTINFO
- Our program can read the 495 (customer number) from the URL itself.

58

# Apache 2.4 Update



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

Newer syntax:

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

Older syntax:

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

If you are using an older release, use this second syntax.

59

# Edit Configuration File



The screenshot shows the IBM Web Administration interface for an Apache server named 'MYDEMO'. On the left is a navigation tree with categories like 'Server Properties', 'Request Processing', 'Security', 'Proxy', 'Tools', and 'Domino Application Server'. The 'Tools' section is expanded, and 'Edit Configuration File' is highlighted with a red circle. A red arrow points from this menu item to the main content area. The main content area has a title 'Manage Apache server "MYDEMO" - Apache/2.2.11 (i5)' and a sub-header 'Demonstrate RPG Web Services'. Below this is a welcome message and several sections of text providing instructions and tips for managing the server. A text box is overlaid on the right side of the main content area, containing instructions to scroll to the 'Tools' section and use 'edit configuration file' to enter Apache directives, along with a tip about using 'Display configuration file' to check for errors.

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.

60

# Alternate Recipe



The last slide shows how to make /cust always do a call restful/custinfo.

But, perhaps you'd rather not have to key a separate Apache configuration for each restful web service you want to run? There are pros and cons to this:

- Don't have to stop/start server to add new service.
- Any program left in RESTFUL library can be run from outside. If the wrong program gets compiled into this library, it could be a security hole.

```
ScriptAlias /cust /qsys.lib/restful.lib/custinfo.pgm

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

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

```
http://ibmi.example.com/rest/custinfo/495
```

61

# Add Custom Directives



Server: MYDEMO - Apache Server area: Global configuration

```
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
    Deny From all
</Directory>
<Directory /www/mydemo/htdocs>
    Order Allow,Deny
    Allow From all
</Directory>

# Scott's RESTFUL webservices

ScriptAlias /cust /qsys.lib/skwebsrv.lib/custinfo.pgm
<Directory /qsys.lib/skwebsrv.lib>
    Order Allow,Deny
    Allow From All
</Directory>

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

OK Apply Cancel

Scroll down to the bottom of the file.

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

62

# Start New Apache Server



The screenshot shows the IBM Web Administration console for a WebSphere environment. The 'Server Properties' tree on the left has 'Display Configuration File' selected. The main pane shows the configuration file content for 'MYDEMO'. A red arrow points from the 'Display Configuration File' button in the left pane to the 'Display Configuration File' button in the top toolbar. Another red arrow points from the 'Display Configuration File' button in the top toolbar to the 'Start' button (green play icon) in the top toolbar.

```
1 # Configuration originally created by Create HTTP Server
2 Listen *:8543
3 DocumentRoot /www/mydemo/htdocs
4 TraceEnable Off
5 Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNo
6 LogFormat "%h %T %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" com
7 LogFormat "%{Cookie}n \"%r\" %t\" cookie
8 LogFormat "%{User-agent}i\" agent
9 LogFormat "%{Referer}i -> %U\" referer
10 LogFormat "%h %l %u %t \"%r\" %>s %b\" common
11 CustomLog logs/access_log combined
12 LogMaint logs/access_log 7 0
13 LogMaint logs/error_log 7 0
14 SetEnvif "User-Agent" "Mozilla/2" nokeepalive
15 SetEnvif "User-Agent" "JDK/1.0" force-response-1.0
16 SetEnvif "User-Agent" "Java/1.0" force-response-1.0
17 SetEnvif "User-Agent" "RealPlayer 4.0" force-response-1.0
18 SetEnvif "User-Agent" "MSIE 4.0b2;" nokeepalive
19 SetEnvif "User-Agent" "MSIE 4.0b2;" force-response-1.0
20 <Directory />
21 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)

# RESTful Example



## Easier way to think of REST

- input can come from the URL, cookies, headers or an uploaded document
- if a document – it can be anything (XML, JSON or something else...)
- output has no standard... can be anything (but usually is XML or JSON)

For example, you might have a web service that takes a customer number as input and returns that customer's address.

Input

```
GET http://i.scottklement.com:8500/cust/495
```

Output

```
{
  "CUSTNO": 495,
  "NAME": "Acme Foods",
  "STREET": "1100 NW 33rd Street",
  "CITY": "Minneapolis",
  "STATE": "MN",
  "POSTAL": "43064-2121"
}
```

# 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 known as Common Gateway Interface (CGI)

- The URL you were called from is available via the `REQUEST_URI` env. var
- If a document 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)

- `QtmhRdStin` ← reads standard input
- `getenv` ← retrieves an environment variable.
- `QtmhWrStout` ← writes data to standard output.

Or we can use the YAJL toolkit, which is free (open source) and will handle the standard input and output for us when it interprets a JSON document.

65

# DIY REST Example (1 of 2)



```
Ctl-Opt OPTION(*SRCSTMT: *NODEBUGIO) DFTACTGRP(*NO);

Dcl-F CUSTFILE Usage(*Input) Keyed prefix('CUST. ');
dcl-ds CUST ext extname('CUSTFILE') qualified end-ds;

Dcl-PR getenv Pointer extproc('getenv');
  var Pointer value options(*string);
End-PR;

dcl-s custno like(CUST.custno);
Dcl-S pos int(10);
Dcl-S uri varchar(1000);
Dcl-S json varchar(1000);
Dcl-C ID1 '/cust/';
Dcl-C ID2 '/custinfo/';

dcl-ds failure qualified;
  error varchar(100);
end-ds;
```

getenv lets us retrieve an environment variable – the URL will be in the `REQUEST_URI` variable.

We can generate JSON from a DS using RPG's `DATA-GEN` opcode.

So the `CUST DS` can be output directly if all is well.

If there's an error, we'll put the message in the `FAILURE DS`

66

## DIY REST Example (2 of 2)



```
uri = %str(getenv('REQUEST_URI'));  
  
monitor;  
  pos = %scan(ID1: uri) + %len(ID1);  
  custno = %int(%subst(uri:pos));  
on-error;  
  failure.error = 'Invalid URI';  
  DATA-GEN failure %DATA(json) %GEN( 'YAJLDTAGEN'  
    : '{ "http status": 500, "write to stdout": true }');  
  return;  
endmon;  
  
chain custno CUSTFILE;  
if not %found;  
  failure.error = 'Unknown customer number';  
  DATA-GEN failure %DATA(json) %GEN( 'YAJLDTAGEN'  
    : '{ "http status": 500, "write to stdout": true }');  
  return;  
endif;  
  
DATA-GEN cust %DATA(json) %GEN( 'YAJLDTAGEN'  
  : '{ "http status": 200, "write to stdout": true }');  
return;
```

REQUEST\_URI will contain  
http://x.com/cust/495

Custno is everything after /cust/ in the URL

If an error occurs, generate a JSON document from the FAILURE DS.

If no errors, generate it from the CUST DS.

"write to stdout" causes YAJL to write the result to Apache.

"http status" lets us set the HTTP status code to 200 for success, 500 for error.

67

## Changes To Use W/Alt Recipe



To use the alternate Apache config (ScriptAliasMatch) change this code:

```
monitor;  
  pos = %scan(ID1: uri) + %len(ID1);  
  custno = %int(%subst(uri:pos));  
  . . .
```

To this... it now works on anything after /cust/ or /custinfo/ in the URI

```
Dcl-C ID1      '/cust/';  
Dcl-C ID2      '/custinfo/';  
.  
.  
monitor;  
  select;  
  when %scan(ID1: uri) > 0;  
    pos = %scan(ID1: uri) + %len(ID1);  
  when %scan(ID2: uri) > 0;  
    pos = %scan(ID2: uri) + %len(ID2);  
  other;  
    pos = 0;  
  ends1;  
  custno = %int(%subst(uri:pos));  
  . . .
```

68

## About Testing and Consuming DIY



There's nothing special about testing a DIY example. You call it the same as any other (REST) web service – just use SoapUI (or a similar tool like Postman), just as we did with the IWS example.

You'll notice that using the HTTP server isn't much harder than using the IWS was – the code is nearly as simple (thanks to DATA-GEN and YAJL)

The DIY method is much more versatile, however, and performs better.

69

## This Presentation



You can download a PDF copy of this presentation as well as other related materials from:

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

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

# Thank you!

70