Integrated web services server for IBM i updates

Nested array support, parameter ordering, and much more!

Nadir Amra

August 04, 2015
(First published August 04, 2015)

The integrated web services server for IBM i allow users to expose Integrated Language Environment (ILE) programs and service programs as SOAP or Representational State Transfer (REST) web services. In July 2015, a number of enhancements has been released. This article discusses each of the enhancements in the integrated web services server.

Introduction

For several years now, IBM i users have had the ability to deploy ILE programs and services programs as web services based on the SOAP protocol using the integrated web services server for IBM i support that is part of the operating system. It has been just over a year that users were given the ability to deploy ILE programs and service programs as RESTful web services.

In July of 2015, a bunch of enhancements were made to eliminate some of the nuances and limitations when deploying an ILE program object as a web service to an integrated web services server. The following updates have been made:

- Support nested output arrays
- Enable improved processing of very large output character fields
- Preserve case sensitivity of identifiers
- Preserve field ordering
- Allow RESTful services to return user-defined media types
- Allow for new transport metadata values to be passed to web service
- Install web service script updated for SOAP services
- Allow Java™-based web services

In order to get the updates, you will need to load the latest Hypertext Transfer Protocol (HTTP) group program temporary fix (PTF). Table 1 lists the HTTP group PTFs that are needed for each of the supported releases of the IBM i operating system.

Table 1. Software prerequisites

IBM i release	HTTP group PTF

i 7.2	SF99713 (level 9 or greater)
i 7.1	SF99368 (level 35 or greater)
i 6.1	SF99115 (level 45 or greater)

Note that all the updates discussed in the article applies to the integrated web services server version 2.6 or later except the install web service script enhancement, which applies to all versions of the server. In addition, the REST updates do not apply to the IBM i 6.1 release because REST is not supported on 6.1.

Support nested output arrays

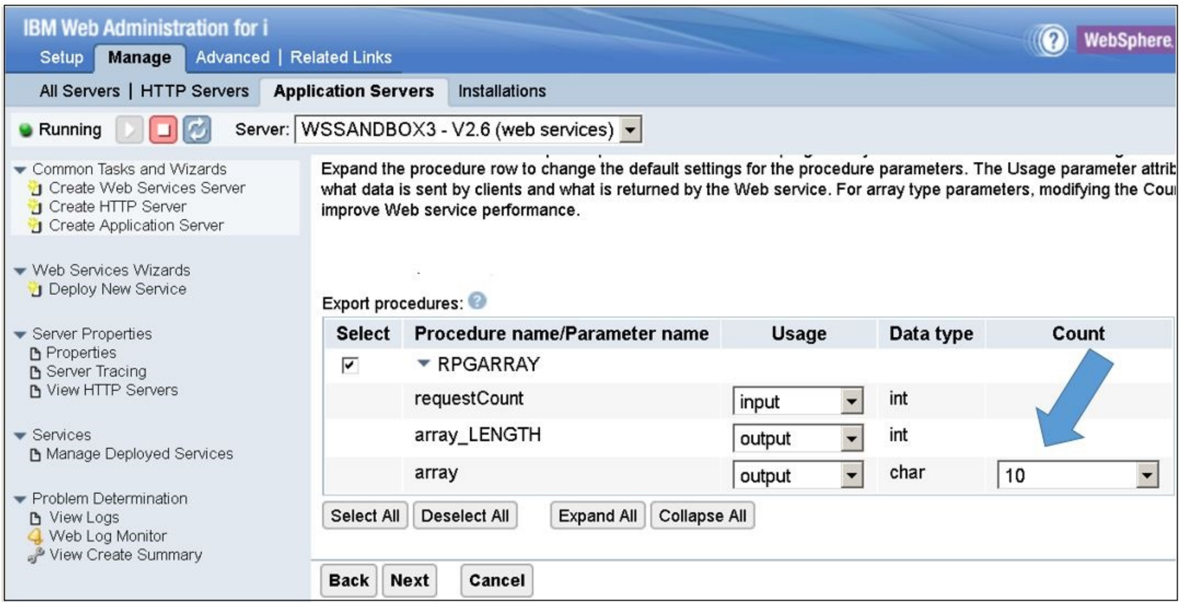
It has always been the case that during the web service deployment of an ILE program object (that is, program or service program) you could designate a parameter that is of type *integer* as the variable that would indicate how many elements are to be returned in an output parameter where the output parameter is an array. For example, Listing 1 shows a simple RPG procedure that has three parameters: `requestCount`, `array_LENGTH`, and `array`. The first parameter, `requestCount`, is an input parameter that indicates how many elements should be returned. The second parameter, `array_LENGTH`, is an output parameter that indicates how many elements were actually returned. And the third parameter, `array`, is an output parameter that is an array of size 10 that contains the elements to be returned.

Listing 1. RPG procedure with array as a parameter

```
h nomain PGMINFO(*PCML:*MODULE:*DCLCASE)

D rpgarray        Pr
D  requestCount...
D                              10i 0
D  array_LENGTH                10i 0
D  array                       20    DIM(10)

P rpgarray        B                  EXPORT
D                 Pi
D  requestCount...
D                              10i 0
D  array_LENGTH                10i 0
D  array                       20    DIM(10)

D i               S            10i 0

 /Free
   if (requestCount < 0 or requestCount > 10);
     requestCount = 0;
   endif;

   array_LENGTH = requestCount;
   clear array;

   for i = 1 to requestCount;
    array(i)          = 'element-' + %CHAR(i);
   endfor;

  return;
 /End-Free
p rpgarray        e
```

When you deploy the service program, a web panel (see Figure 1) that requires you to designate which parameters are input and which parameters are output is shown.

Figure 1. Deploy a new service - Export procedures to externalize as a web service



Server: WSSANDBOX3 - V2.6 (web services)

Expand the procedure row to change the default settings for the procedure parameters. The Usage parameter attrib
what data is sent by clients and what is returned by the Web service. For array type parameters, modifying the Cou
improve Web service performance.

Export procedures:

Select	Procedure name/Parameter name	Usage	Data type	Count
✓	RPGARRAY			
	requestCount	input	int	
	array_LENGTH	output	int	
	array	output	char	10

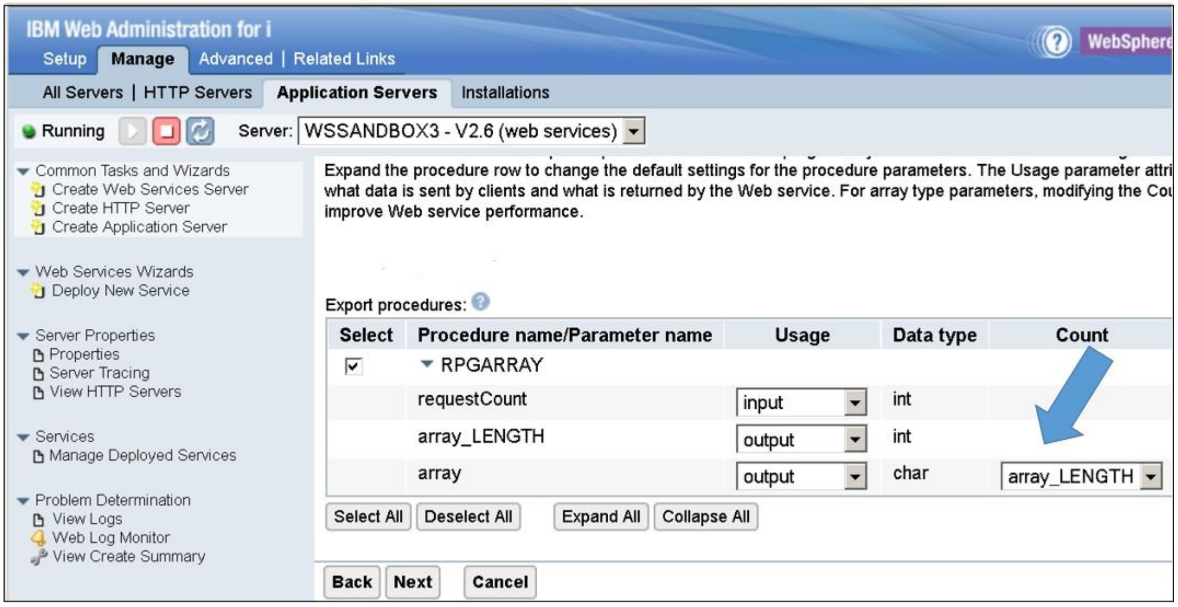
As you can tell by Figure 1, there is a **Count** column where you have the ability to indicate the parameter that contains the actual number of elements in the array. By default, it is set to the maximum size of the array. If we deployed the service program as-is and invoked the web service, the XML document returned would include all the elements in the array, even though we just requested 2 to be returned (see Listing 2).

Listing 2. SOAP response with count not set

```
<soap:Body>
      <ns2:rpgarrayResponse xmlns:ns2="http://rpgarray.wsbeans.iseries/">
         <return>
            <array>element-1</array>
            <array>element-2</array>
            <array/>
            <array/>
            <array/>
            <array/>
            <array/>
            <array/>
            <array/>
            <array/>
            <array_LENGTH>2</array_LENGTH>
         </return>
      </ns2:rpgarrayResponse>
</soap:Body>
```

Now if we redeployed the service program and designate a parameter that indicates the number of actual elements in the output array, as shown in Figure 2, you would have seen a response that did not include the empty elements shown in Listing 2.

Figure 2. Deploy new service - Export procedures to externalize as a web service with count set



Expand the procedure row to change the default settings for the procedure parameters. The Usage parameter attri
what data is sent by clients and what is returned by the Web service. For array type parameters, modifying the Co
improve Web service performance.

Export procedures:

Select	Procedure name/Parameter name	Usage	Data type	Count
✓	RPGARRAY			
	requestCount	input	int	
	array_LENGTH	output	int	
	array	output	char	array_LENGTH

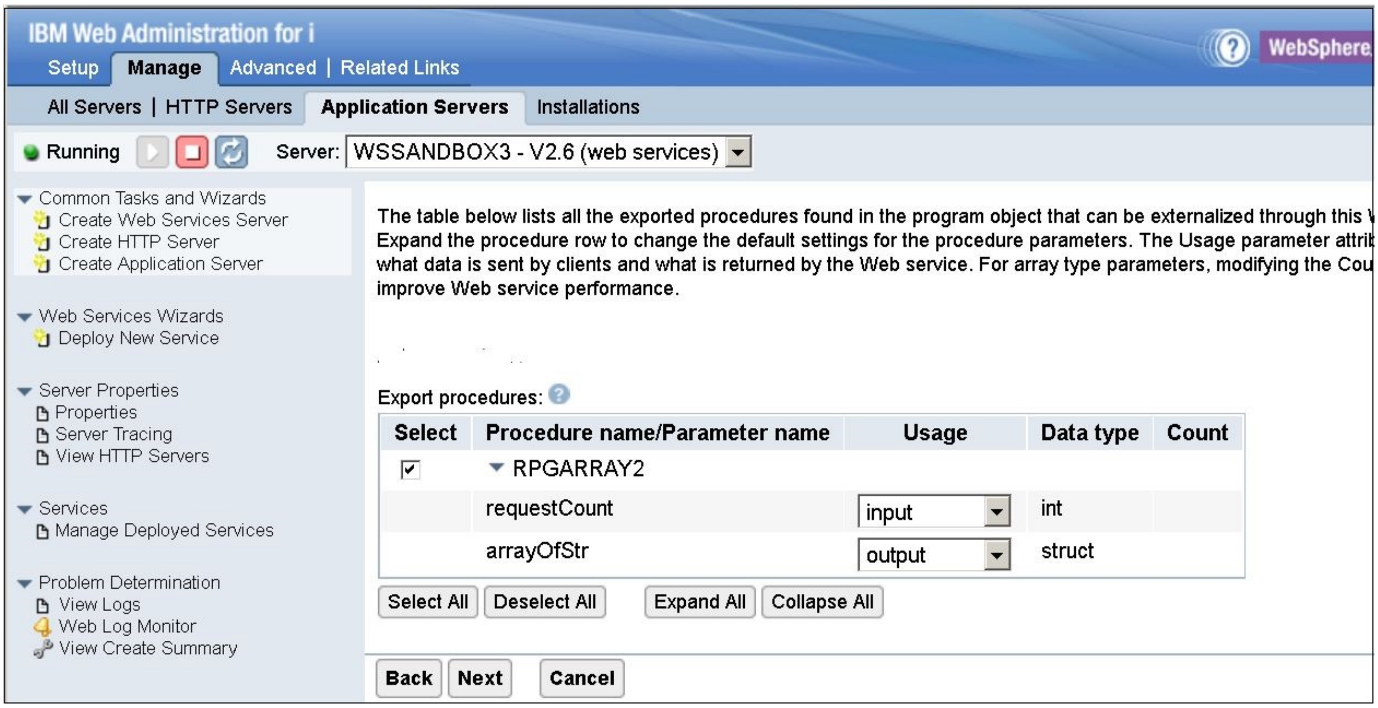
The caveat is that you could only do this for program or procedure parameters. Let us look at an example in Listing 3, which is a slight modification of the example shown in Listing 1.

Listing 3. RPG procedure with array as a field in a data structure

```
h nomain PGMINFO(*PCML:*MODULE:*DCLCASE)

D arrayOfStrDS    DS                  qualified template
D  array_LENGTH                 10i 0
D  array                        20    DIM(10)


D rpgarray2       Pr
D  requestCount...
D                               10i 0
D  arrayOfStr                         likeds(arrayOfStrDS)

P rpgarray2       B                   EXPORT
D                 Pi
D  requestCount...
D                               10i 0
D  arrayOfStr                         likeds(arrayOfStrDS)

D i               S             10i 0

 /Free
   if (requestCount < 0 or requestCount > 10);
     requestCount = 0;
   endif;

   arrayOfStr.array_LENGTH = requestCount;
   clear arrayOfStr.array;

   for i = 1 to requestCount;
    arrayOfStr.array(i)          = 'element-' + %CHAR(i);
   endfor;

   return;
 /End-Free
p rpgarray2       e
```

The change that has been made is to make the `array_LENGTH` and `array` subfields in a structure, `arrayOfStrDS`, and the output parameter, `arrayOfStr`, is defined to be a parameter of type `arrayOfStrDS`. Now when we try to deploy the service program, there is no way to set the count field because the array is not a first-level parameter, as shown in Figure 3.

Figure 3. Deploy new service – Export procedures to externalize as a web service and output parameter is a structure



The table below lists all the exported procedures found in the program object that can be externalized through this \
Expand the procedure row to change the default settings for the procedure parameters. The Usage parameter attrib
what data is sent by clients and what is returned by the Web service. For array type parameters, modifying the Cou
improve Web service performance.

Export procedures:

Select	Procedure name/Parameter name	Usage	Data type	Count
✓	RPGARRAY2			
	requestCount	input	int	
	arrayOfStr	output	struct	

If we invoked the web service, the response would include empty elements, as shown in Listing 4.

Listing 4. SOAP response output structure and count not set

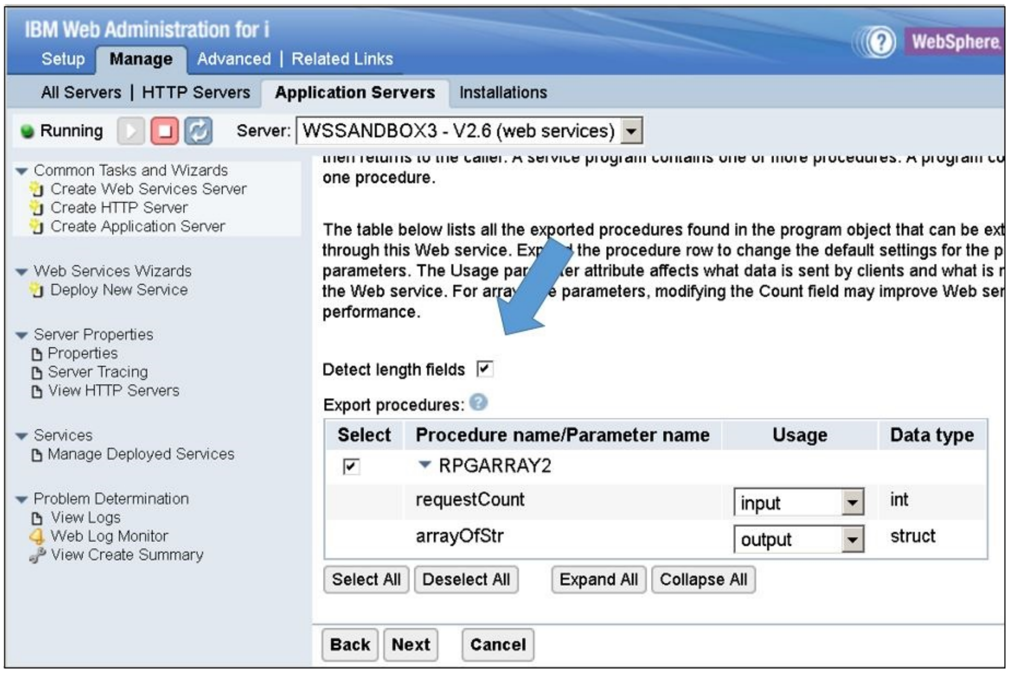
```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:rpgarray2Response xmlns:ns2="http://rpgarray2.wsbeans.iseries/">
         <return>
            <arrayOfStr>
               <array>element-1</array>
               <array>element-2</array>
               <array/>
               <array/>
               <array/>
               <array/>
               <array/>
               <array/>
               <array/>
               <array/>
               <array_LENGTH>2</array_LENGTH>
            </arrayOfStr>
         </return>
      </ns2:rpgarray2Response>
    </soap:Body>
```

```
  </soap:Envelope>
```

Not very nice to say the least. Well, that has now been resolved!

In the panel where you indicate which parameters are input and which parameters are output is a new checkbox field (see Figure 4), **Detect length fields**, that is selected by default.

Figure 4. Deploy new service with Detect length fields selected



Server: WSSANDBOX3 - V2.6 (web services)

then returns to the caller. A service program contains one or more procedures. A program co
one procedure.

The table below lists all the exported procedures found in the program object that can be ext
through this Web service. Exp d the procedure row to change the default settings for the p
parameters. The Usage par er attribute affects what data is sent by clients and what is r
the Web service. For array e parameters, modifying the Count field may improve Web ser
performance.

Detect length fields ☑

Export procedures:

Select	Procedure name/Parameter name	Usage	Data type
☑	RPGARRAY2		
	requestCount	input	int
	arrayOfStr	output	struct

When **Detect length fields** is selected, it will be assumed that any integer (int) field that immediately precedes an array field with the same name as the array field appended with `_LENGTH` is a length field that will be used to indicate the actual number of elements in the array. In this example, the array field identifier is named `array`, then the field length for the array would be `array_LENGTH`. An added benefit is that the length field is hidden from the client and is not returned in the client response. Listing 5 shows the response when **Detect length fields** is selected.

Listing 5. SOAP response output structure with Detect length fields selected

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <ns2:rpgarray2Response xmlns:ns2="http://rpgarray2.wsbeans.iseries/">
         <return>
            <arrayOfStr>
               <array>element-1</array>
               <array>element-2</array>
            </arrayOfStr>
         </return>
      </ns2:rpgarray2Response>
   </soap:Body>
</soap:Envelope>
```

Note: If **Detect length fields** is selected, you will only be able to specify input or output parameters – you will not be able to specify inputoutput parameters. In addition, the preservation of parameter order and case sensitivity of identifiers, which we will talk about later in this article, is also enabled.

Enable improved processing of very large output character fields

Length field support has been extended to character fields. Output character fields that are very large (I am talking about 1000s of bytes) take time to process because the determination of the size of the string to return is done by traversing the field a byte at a time, from right to left, looking for the first non-blank character.

To improve the processing of these large character fields, you can specify an integer (int) field length that immediately precedes the character field with the same name as the character field appended with `_LENGTH` (similar to what is done when processing output arrays with field lengths). If the field length exists and **Detect length fields** is selected, the traversing of the field to determine the size is not performed, improving performance. In addition, the length field is hidden from the client and is not returned in the client response.

Preserve case sensitivity of identifiers

Identifier Generation

A new RPG enhancement has been released for IBM i releases 7.1 and 7.2 that allows you to control the identifier case of parameter names. See the following PTFs for details:

- SI555317.2
- SI554427.2
- SI554407.1

In the past, the compiler generated a Program Call Markup Language (PCML) document and stored it in the ILE module object with identifiers entirely uppercased. Thus identifiers in SOAP and REST requests and responses had to be uppercased. In the latter half of 2014, an enhancement was made in the RPG compiler that preserved the identifier case when generating the PCML document. This allowed integrated web services to preserve identifier case to a certain extent. Identifiers for XML element names and JSON field names were generated based on the procedure name, the structure name, and the field names within a structure. The first character in the identifier is changed from uppercase to lowercase unless the first two characters are in uppercase, in which case the identifier is left alone.

If the **Detect length fields** check box is selected, the identifier name is used as it is defined in the ILE program object. For example, if the identifier starts with an uppercase character followed by a lowercase character, then the identifier in the XML or JSON document will also start with an uppercase character followed by a lowercase character. If the **Detect length fields** is not selected, then the identifier will start with a lowercase character.

Preserve field ordering

Prior to the latest integrated web services updates, when deploying an ILE program or service program, the ordering of parameter fields and fields within structures was not preserved, and in most cases the ordering is based on alphabetic ordering (and sometimes it is based on something that I cannot even fathom!). Take for example the RPG procedure shown in Listing 6.

Listing 6. RPG procedure to illustrate field ordering

```
h nomain PGMINFO(*PCML:*MODULE:*DCLCASE)

D ordertest       Pr
D  oneParam                     10i 0
D  twoParam                     10i 0
D  threeParam                   10i 0

P ordertest       B                   EXPORT
D                 Pi
D  oneParam                     10i 0
D  twoParam                     10i 0
D  threeParam                   10i 0

 /Free
   oneParam = 1;
   twoParam = 2;
   threeParam = 3;
   return;
 /End-Free
p orderTest       e
```

If the service program is deployed and all three parameters are designated as output parameters, you might think that the XML response would include the three parameters in the following sequence: `oneParam`, `twoParam`, and `threeParam` . However, the SOAP XML response that is returned is shown in Listing 7.

Listing 7. SOAP XML response – unordered fields

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <a:ordertestResponse xmlns:a="http://ordertest.wsbeans.iseries/">
         <return>
            <twoParam>2</twoParam>
            <oneParam>1</oneParam>
            <threeParam>3</threeParam>
         </return>
      </a:ordertestResponse>
   </soap:Body>
</soap:Envelope>
```

If the **Detect length fields** check box is selected, then the ordering is preserved, and the response you will get is shown in Listing 8.

Listing 8. SOAP XML response – ordered fields

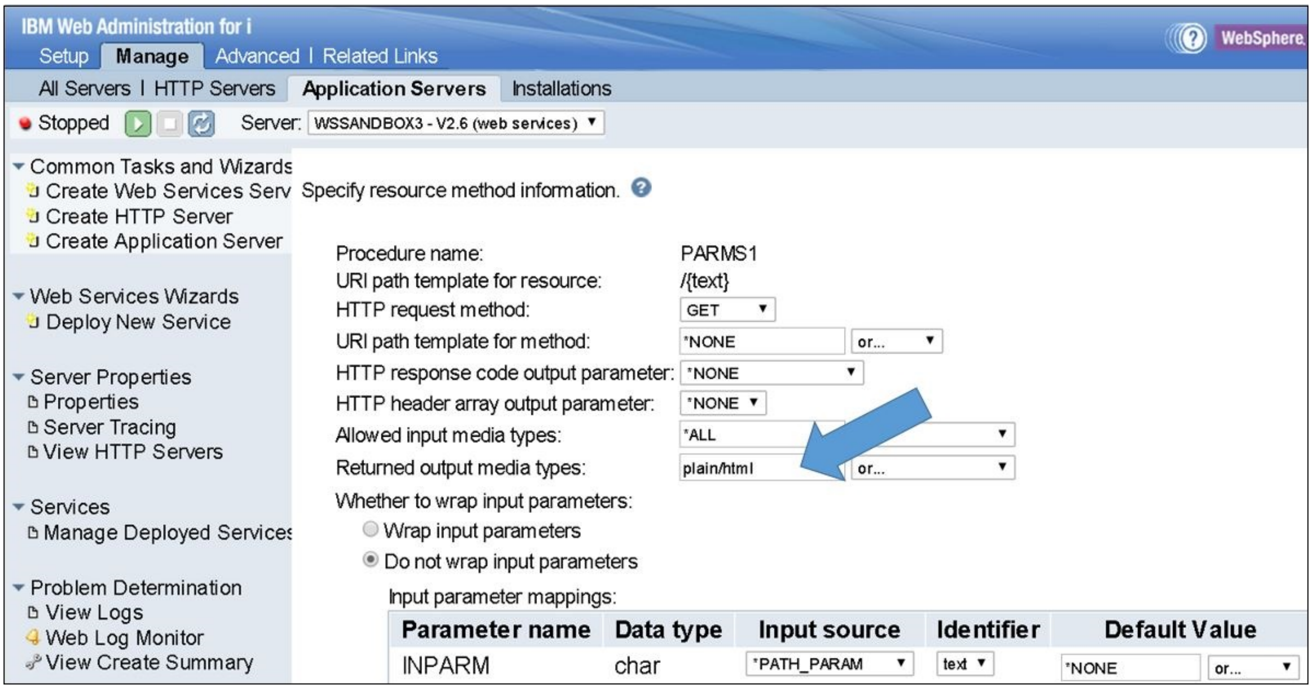
```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <a:ordertestResponse xmlns:a="http://ordertest.wsbeans.iseries/">
         <return>
            <oneParam>1</oneParam>
            <twoParam>2</twoParam>
            <threeParam>3</threeParam>
         </return>
      </a:ordertestResponse>
   </soap:Body>
</soap:Envelope>
```

Allow RESTful services to return user-defined media types

Support has been added to allow RESTful services to return user-defined content. Before this enhancement, the response that is returned could either be XML or JSON content.

To return a user-defined media type, there must be one output parameter (or two if **Detect length fields** is selected and one parameter is a field length for the other parameter which must be of type character). The output parameter must be a primitive type such as integer, char, float, and so on. In most cases, the output parameter would be of type character. The deploy panel in which REST properties are set has been updated to allow the setting of a user-defined media type. In Figure 2, you can see the panel with `plain/html` being set for the output media type. This means that the program will return a string consisting of an HTML document.

Figure 5. Deploy new service – setting REST properties



Specify resource method information.

Procedure name:	PARMS1
URI path template for resource:	/{text}
HTTP request method:	GET
URI path template for method:	*NONE or...
HTTP response code output parameter:	*NONE
HTTP header array output parameter:	*NONE
Allowed input media types:	*ALL
Returned output media types:	plain/html or...

Whether to wrap input parameters:
( ) Wrap input parameters
(●) Do not wrap input parameters

Input parameter mappings:

Parameter name	Data type	Input source	Identifier	Default Value
INPARM	char	*PATH_PARAM	text	*NONE or...

More than one media type value can be specified by ensuring that each media type is separated by a comma. If more than one is specified, then the ILE program object would need to set the HTTP header content-type field to the appropriate value depending on the content that is being returned. To learn how to set HTTP headers, see Building a REST service with integrated web services server for IBM i, Part 3.

Allow for new transport metadata values to be passed to web service

Before the enhancements, the REMOTE_ADDR (remote address of the client) transport metadata value could be passed to the ILE web service implementation code as an environment variable. Additional transport metadata values have been added:

- QUERY_STRING: Returns the query string that is contained in the request URL after the path.

- REQUEST_METHOD: Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.
- REQUEST_URI: Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.
- REQUEST_URL: Reconstructs the URL the client used to make the request.
- SERVER_NAME: Returns the host name of the server to which the request was sent. It is the value of the part before ":" in the `Host` header value, if any, or the resolved server name or the server IP address.
- SERVER_PORT: Returns the port number to which the request was sent. It is the value of the part after ":" in the `Host` header value, if any, or the server port where the client connection was accepted on.

Note: If the web service was deployed before applying the HTTP group PTF that includes the integrated web services updates, then you will need to redeploy the web service in order for the new transport metadata to be passed to the web service implementation code.

Install web service script updated for SOAP services

The installWebservice.sh QShell script (located in /qibm/proddata/os/webservices/bin) has been enhanced so that you can now deploy an ILE program object and specify parameter usage (that is, input, output, and inputoutput) when deploying a SOAP service. The `-parameterUsage` option is a colon delimited list containing parameter usage values corresponding to the procedures or program to be deployed. For each program or procedure you need to specify the procedure or the program name, followed by a colon, followed by a comma delimited list of usage descriptors ('i' for input, 'o' for output, or 'io' for inputoutput) for each parameter. For example, if a service program contains two procedures, 'PROC1' with two parameters and 'PROC2' with three parameters, then a possible value would be:

```
-parameterUsage PROC1:i,o:PROC2:i,i,io
```

Note that only those procedures listed will be externalized as web service operations. This parameter is optional. If not specified, the default is to use the parameter usage values specified in the PCML associated with the program object.

In addition, you can now indicate whether you want the web service to be deployed as a SOAP 1.1 or SOAP 1.2 web service. The `-serviceType` option is the type of service to be installed. A value of `*SOAP11` or `*SOAP12` indicates that the program object should be installed as a SOAP 1.1 or SOAP 1.2 service, respectively. This parameter is optional. If not specified, the default value of `*SOAP11` is used.

Allow Java-based web services

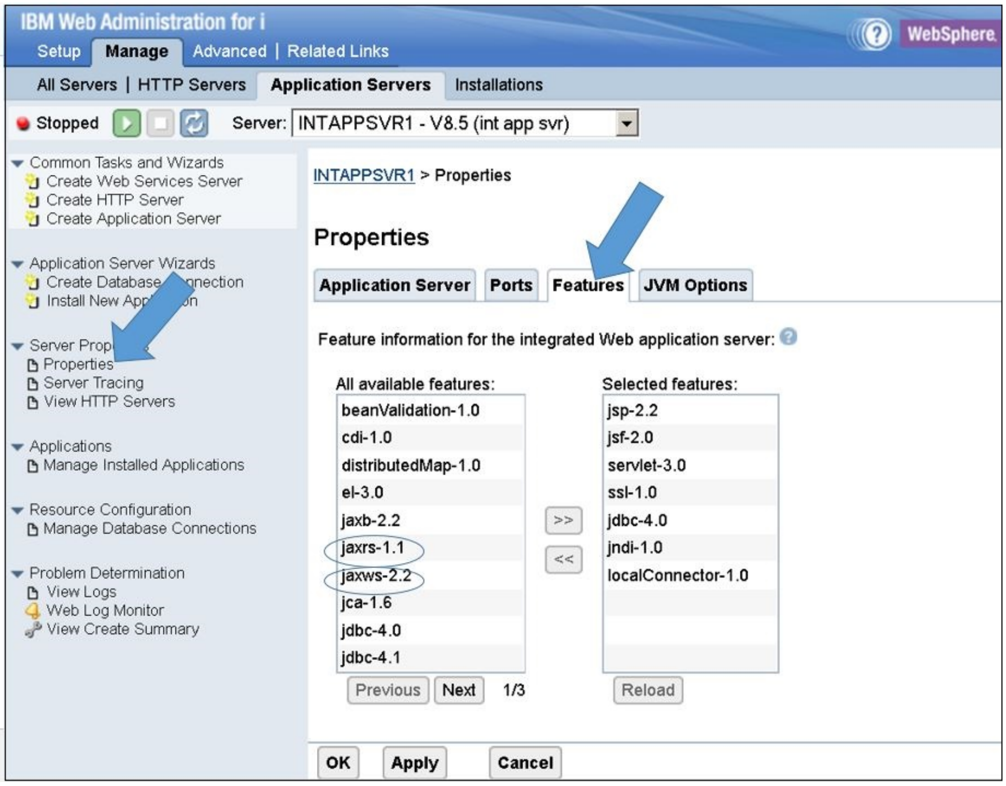
What are features?

The integrated application server version 8.5 is based on IBM WebSphere® Application Server Liberty profile. Liberty has the notion of features, which are units of functionality by which you control the pieces of the runtime environment that are loaded into a particular server.

I have included this bit of information as part of the integrated web services server updates, but in fact it is an enhancement in the integrated application server. The integrated web services server will always be a server that can be used to deploy ILE programs and services programs as web services. However, we realize the need for users to deploy Java web services to a server, so an enhancement has been made to the integrated application server that allows you to enable the JAX-WS and JAX-RS features in an integrated application server. When these features are enabled, you can deploy JAX-WS and JAX-RS web services to the server.

To enable the feature, go to the IBM Web Administration GUI, and select the integrated application server (must be version 8.5 or later). If you click the **Server properties** link in the navigation panel, a page with various tabs will be displayed. On the **Features** tab, you will find various features that can be enabled for the server, as shown in Figure 6.

Figure 6. Integrated application server Features tab



Server: INTAPPSVR1 - V8.5 (int app svr)

INTAPPSVR1 > Properties

Feature information for the integrated Web application server:

All available features: beanValidation-1.0, cdi-1.0, distributedMap-1.0, el-3.0, jaxb-2.2, jaxrs-1.1, jaxws-2.2, jca-1.6, jdbc-4.0, jdbc-4.1

Selected features: jsp-2.2, jsf-2.0, servlet-3.0, ssl-1.0, jdbc-4.0, jndi-1.0, localConnector-1.0

If you want to deploy SOAP web services based on JAX-WS, then you would select the jaxws-2.2 (Java API for XML-Based Web Services 2.2) feature and then click the double-headed arrows pointing to the right, followed by clicking **OK** or **Apply.** You would do the same thing if you want to deploy REST web service based on JAX-RS, except the feature that you would want to add is jaxrs-1.1 (Java API for RESTful Web Services 1.1).

Summary

The integrated web services server support provides a solid foundation for creating and deploying web services based on ILE programs or service programs on the IBM i platform. You can now deploy web services based on Java to integrated application servers too! We're continually trying to improve the integrated web services experience, and we'd love to hear from you.

Resources

- For information about the integrated web services support on IBM i see the product web page.