WSDL2RPG – FAQ

How to Troubleshoot Performance Problems

Status of this document

Date:	09.04.2012
Version:	1.2

Question

What options to I have if calling a Web Service is extremely slow?

Answer

There are several reasons why calling a Web Service might be slow. This document tries to explain how to fix known performance problems and how to analyse these problems at all. It uses example WS0004 to discuss the problem.

This document uses the additional debug information that is been appended to the HTTPAPI debug log with WSDL2RPG v1.15 and higher.

First of all you should try to narrow down the problem. The HTTPAPI debug log is a good starting point, because additional debug messages are added to it starting with WSDL2RPG v1.15. You can enable the debug log like this:

```
CountryInfoServiceSoap_Port_setHttpDebug(
   *ON: '/home/raddatz/wsdl2rpg/ws000401tg.log');
```

Messages added by WSDL2RPG always start with "**" and an ISO formatted timestamp value as shown here:

```
** 2012-03-01-09.47.21.611000: Entering
CountryInfoServiceSoap_ListOfCountryNamesByCode()
```

WSDL2RPG debug messages	
Message	**Action**
`Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()`	Sent almost right at the beginning of the web service operation after the stub has been initialized.
`Sending request to server`	Sent when the request message has been produced and before HTTPAPI sends the message to the server.
`Requesting user name and password`	Sent when the server asked for login credentials.
`Preparing to parse received message`	Sent when the first chunk of the message has been received and is about to be parsed.
`Detected MIME message`	Sent when the server sent a MIME message back to the client.
`Leaving CountryInfoServiceSoap_ListOfCountryNamesByCode()`	Sent at the end of the web service operation.

The time between these messages is a good indicator for how long it took to finish particular steps.

Here is a sample log having all unnecessary information removed:

```
HTTPAPI Ver 1.25beta1 released 2012-02-17
...
** 2012-03-01-10.06.47.857000: Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()
...
** 2012-03-01-10.06.47.874000: Sending request to server
http_persist_open(): entered
...
sendraw(): entered
<?xml version="1.0" ...
recvresp(): entered
HTTP/1.1 200 OK
...
recvdoc(): entered
SetError() #0:
<?xml version="1.0" encoding="utf-8"?>
...
        <m:tCountryCodeAndN
** 2012-03-01-10.06.48.460000: Preparing to parse received message
ame>
...
  </soap:Envelope>
http_close(): entered
...
** 2012-03-01-10.06.48.716000: Leaving CountryInfoServiceSoap_ListOfCountryNamesByCode()
```

Building the request message is slow

Dynamic Arrays – Initial Array Size

In case your Web Service stub was build for using dynamic arrays you should check the number of items that are added to the arrays. By default `MemoryManager_createArray()` creates the arrays for an initial size of 256 elements. When the last element is used, the array is incremented to take the next 256 elements. Given that you want to add thousands of elements incrementing the array over and over takes lots of time and you may want to increase the initial size of the array like this:

```
MemoryManager_createArray(
   uuid: %size(newItem_of_string_A1): 'item'
   : *omit: 2000); // Initial size of the array
```

The messages of interest are:

```
Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()

Sending request to server
```

Sample log:

```
** 2012-03-01-10.06.47.857000: Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()
...
** 2012-03-01-10.06.47.874000: Sending request to server
```

In this case there were no problems because it took only 17 milliseconds to produce the request message.

Solution: Specify a reasonable initial size for dynamic arrays.

Dynamic Arrays – Using Symbolic Names

In case your Web Service stub was build for using dynamic arrays you should carefully decide whether or not you need to use a symbolic name instead of the array handle. WSDL2RPG uses a map (list of key/value pairs) to associate the symbolic name (key) with the array handle (value) to keep track of the symbolic names and their associated names. Although the map uses a binary search algorithm it is really slow for a huge number of arrays. You can dramatically improve performance when you disable dynamic names like this:

```
MemoryManager_attachService(uuid: cFalse);
```

The messages of interest are:

```
Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()

Sending request to server
```

Sample log:

```
** 2012-03-01-10.06.47.857000: Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()
...
** 2012-03-01-10.06.47.874000: Sending request to server
```

In this case there were no problems because it took only 17 milliseconds to produce the request message.

Solution: Disable symbolic names whenever it is possible.

Sample: An application that took about 25 minutes to send 20000 by 3 arrays now takes only 12 seconds without using symbolic names.

Request Message Buffer

In case your request message is very long, you may need to increase the memory that the "ManagedMemoryDataSource" uses to cache the request message before it is send to the server. By default the "ManagedMemoryDataSource" uses a 4MB memory cache to store the request message. But when the message exceeds 4MB it is completely written to a temporary stream file and sent to the server from there. Of course using a stream file is slower than memory. In order to increase the cache size you need to change the following statement in the operation module of your Web Service stub:

```
// Setting the cache size to 12MB
// The maximum value is 16776704, which is not exactly 16MB!
// Refer to the RPG Language reference for details.
hDataSource = ManagedMemoryDataSource_new(1024*1024*12);
```

The messages of interest are:

```
Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()

Sending request to server
```

Sample log:

```
** 2012-03-01-10.06.47.857000: Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()
...
** 2012-03-01-10.06.47.874000: Sending request to server
```

In this case there were no problems because it took only 17 milliseconds to produce the request message.

Solution:	Increase the cache size of the ManagedMemoryDataSource for huge request messages that exceed 4MB.

Sending/Receiving the message is slow

The time needed to send the message to the server is the time between the following messages:

```
Sending request to server

Preparing to parse received message
```

Sample log:

```
** 2012-03-01-10.06.47.874000: Sending request to server
http_persist_open(): entered
...
sendraw(): entered
<?xml version="1.0" ...
recvresp(): entered
HTTP/1.1 200 OK
...
recvdoc(): entered
SetError() #0:
<?xml version="1.0" encoding="utf-8"?>
...
        <m:tCountryCodeAndN
** 2012-03-01-10.06.48.460000: Preparing to parse received message
```

Actually the 585 milliseconds of the sample above are not only the time it took to send the message but also the time to receive the first chunk of the response message.

There is no way for WSDL2RPG to be more precise without changing HTTPAPI.

Solution:	This problem is difficult to solve. Check your network infrastructure and if that is fine, contact the web service provider.

Parsing the response message is slow

The time between the following messages indicate how long it took to parse the response message:

```
Preparing to parse received message
```

```
Leaving CountryInfoServiceSoap_ListOfCountryNamesByCode()
```

If the log contains the following message, the MIME parser may also be responsible for the problem:

```
Detected MIME message
```

In case your stub module uses dynamic arrays you may read the chapter about “*Dynamic Arrays*”.

Sample log:

```
** 2012-03-01-10.06.48.460000: Preparing to parse received message
ame>
...
  </soap:Envelope>
http_close(): entered
...
** 2012-03-01-10.06.48.716000: Leaving CountryInfoServiceSoap_ListOfCountryNamesByCode()
```

In this sample it roughly took 256 milliseconds to parse the response message.

Solution:	Refer to the chapter about dynamic arrays if your Web Service stub uses them.

Getting Additional Debug Messages

In “verbose” mode the debug log shows additional messages, e.g. showing the time that it took to parse each chunk of the response message. Sometimes that might be helpful to get another idea of what the problem might be.

You can enable “verbose” mode as shown below:

```
    CountryInfoServiceSoap_Port_setHttpDebug(
       *ON: '/home/raddatz/wsdl2rpg/ws000401tg.log': cTrue);
```

Sample log with additional messages in green:

```
HTTPAPI Ver 1.25beta1 released 2012-02-17
...
** 2012-03-01-13.03.51.917000: Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()
...
** 2012-03-01-13.03.51.944000: Producing request message
** 2012-03-01-13.03.51.954000: Finished request message
** 2012-03-01-13.03.51.954000: Preparing to send message
** 2012-03-01-13.03.51.954000: Sending request to server
http_persist_open(): entered
...
sendraw(): entered
<?xml version="1.0" ...
recvresp(): entered
HTTP/1.1 200 OK
...
recvdoc(): entered
SetError() #0:
<?xml version="1.0" encoding="utf-8"?>
...
        <m:tCountryCodeAndN
** 2012-03-01-13.03.52.577000: Entering HTTP_receiveResponse()
** 2012-03-01-13.03.52.577000: Preparing to parse received message
** 2012-03-01-13.03.52.581000: Leaving HTTP_receiveResponse()
ame>
...
          <m:sISOCode>BH</m:sISOCode>
** 2012-03-01-13.03.52.752000: Entering HTTP_receiveResponse()
** 2012-03-01-13.03.52.760000: Leaving HTTP_receiveResponse()

          <m:sName>Bahrain</m:sName>
...
        </m:tCountryCodeAndName>

** 2012-03-01-13.03.52.761000: Entering HTTP_receiveResponse()
** 2012-03-01-13.03.52.780000: Leaving HTTP_receiveResponse()
    <m:tCountryCodeAndName>
...
          <m:sName>Moldova, Republic of</m:sName>

** 2012-03-01-13.03.52.781000: Entering HTTP_receiveResponse()
** 2012-03-01-13.03.52.799000: Leaving HTTP_receiveResponse()
   </m:tCountryCodeAndName>
...
          <m:sISOCode>SG</m:sISOCo
** 2012-03-01-13.03.52.800000: Entering HTTP_receiveResponse()
** 2012-03-01-13.03.52.818000: Leaving HTTP_receiveResponse()
de>
...
  </soap:Envelope>
** 2012-03-01-13.03.52.819000: Entering HTTP_receiveResponse()
** 2012-03-01-13.03.52.835000: Leaving HTTP_receiveResponse()
http_close(): entered
...
** 2012-03-01-13.03.52.836000: Leaving CountryInfoServiceSoap_ListOfCountryNamesByCode()
```

Additional WSDL2RPG debug messages	
Message	**Action**
`Entering CountryInfoServiceSoap_ListOfCountryNamesByCode()`	Sent almost right at the beginning of the web service operation after the stub has been initialized.
`Producing request message`	Sent when the stub start producing the request message.
`Finished request message`	Sent when the request message has been produced.
`Preparing to send message`	Sent when the stub prepares to send the message.
`Sending request to server`	Sent when the request message has been produced and before HTTPAPI sends the message to the server.
`Requesting user name and password`	Sent when the server asked for login credentials.
`Entering HTTP_receiveResponse()`	Sent when HTTPAPI passes a chunk of the response message to WSDL2RPG.
`Preparing to parse received message`	Sent when the first chunk of the message has been received and is about to be parsed.
`Detected MIME message`	Sent when the server sent a MIME message back to the client.
`Leaving HTTP_receiveResponse()`	Sent when WSDL2RPG has parsed a chunk of the response message.
`Leaving CountryInfoServiceSoap_ListOfCountryNamesByCode()`	Sent at the end of the web service operation.

Your comments are important to me! Please send me your comments about this FAQ. I will greatly appreciate it.

thomas.raddatz@tools400.de