

Copyright 2006 Krengel Technologies Inc.

Author: Aaron Bartell

Program: EXAMPLE, RXS3

```
*****
// @Author: Aaron Bartell
// @Creation Date:
// @Desc: Read in xml and parse the PostAdr document. Store the values locally and
//        respond with whether or not the processing was successful.
// @Notes: SOAP is not used for this web service.
*****
H dftactgrp(*no) bnddir('RXSBND')

/copy rxs,RXScp

//-----
// Local Prototypes
//-----
D parse          pr
D compose        pr

D allHandler     pr
D pType          value like(RXS_Type)
D pxPath         value like(RXS_XPath)
D pData          value like(RXS_XmlData)
D pDataLen       value like(RXS_Length)

D errHandler     pr
D pCurLine      10i 0 value
D pCurCol       10i 0 value
D pErrStr        1024a value varying

D errRsp         pr

D gAllPrcPtr     s          *   procptr inz(%paddr(allHandler))
D gErrPrcPtr     s          *   procptr inz(%paddr(errHandler))

//-----
// Global Variables
//-----
D gPostAdr       ds          qualified inz
D residential    n
D title          5a
D firstName      15a varying
D lastName       15a varying
D street         15a
D cty            10a
D state          2a
D zip            5a
D phone          12a dim(2)

D gPhnCnt        s          10i 0 inz(0)
D gError         ds          likeds(RXS_Error)
D gXml           s          65535a varying
D gSndHdr        s          n
```

```
-----  
// Mainline: First initialize all necessary fields. Second, read in the xml document and place  
// it in gXml. Next call the parse() sub-procedure which will prep the parser and  
// invoke it to process the xml stored in gXml.  
//  
// After parsing check for errors in gError.code to see if processing should  
// continue.  
//  
// Once the document has been parsed and the data placed in the appropriate DB2  
// tables a response can be composed that will be sent to the requester.  
//  
// If any errors occur ANYWHERE it will be captured with the 'monitor' clause and  
// a corresponding error message will be written to standard out.  
-----
```

```
/free
```

```
monitor;  
  clear gError;  
  gSndHdr = *on;  
  
  gXml = RXS_readStdIn();  
  
  parse();  
  
  if gError.code <> *blanks;  
    errRsp();  
    return;  
  endif;  
  
  //  
  // Insert business logic to write data residing in gPostAdr DS to DB2 tables.  
  //  
  
  compose();  
  
on-error;  
  RXS_stdOutError('error': RXS_catchError(): *on);  
endmon;  
*inlr = *on;
```

```
/end-free
```

```
-----  
// @Author: Aaron Bartell  
// @Created: 2005-08-03  
// @Desc: Setup event handlers that will get called each time an element content or attribute  
// is encountered. Note that ALL elem content and attributes encountered will have  
// events generated for them. This is an ease-of-use approach that saves you from  
// having to setup event notification for each element content and attribute manually.  
//  
// Next call the parser with the xml received in on the request. Tell the parser it is  
// parsing a variable's value by specifying RXS_VAR. The other option is to parse a  
// file in the IFS by specifying RXS_STMF. The last parm is the procedure pointer of  
// the local procedure that should be notified in the event a parsing error occurs.  
//  
// If any errors occur the 'monitor' clause will catch it and place it in the gError  
// variable.  
-----
```

```
P parse          b  
D parse          pi  
/free
```

```
// Example XML that is being parsed:  
//<PostAdr residential="true">  
// <name title="Mr.">  
// <first>Aaron</first>  
// <last>Bartell</last>  
// </name>  
// <street>123 Center Rd</street>  
// <cty>Mankato</cty>  
// <state>MN</state>  
// <zip>56001</zip>  
// <phone>123-123-1234</phone>  
// <phone>321-321-4321</phone>  
//</PostAdr>
```

```
monitor;  
  RXS_allElemContentHandler(gAllPrcPtr);  
  RXS_allAttrHandler(gAllPrcPtr);  
  RXS_parse(gXml: RXS_VAR: gErrPrcPtr);  
on-error;  
  gError = RXS_catchError();  
endmon;
```

```
/end-free  
P          e
```

```

-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: This local sub procedure will be called for each element content and attribute event
//         that occurs during the parsing of the PostAdr document. Based on the event this
//         local sub procedure will place the value in the appropriate global variable.
-----
P allHandler      b
D allHandler      pi
D pType           value like(RXS_Type)
D pxPath          value like(RXS_XPath)
D pData          value like(RXS_XmlData)
D pDataLen       value like(RXS_Length)
/free

select;

when pxPath = '/PostAdr@residential';
  if pData = 'true';
    gPostAdr.residential = *on;
  else;
    gPostAdr.residential = *off;
  endif;

when pxPath = '/PostAdr/name@title';
  gPostAdr.title = pData;

when pxPath = '/PostAdr/name/first/';
  gPostAdr.firstName = pData;

when pxPath = '/PostAdr/name/last/';
  gPostAdr.lastName = pData;

when pxPath = '/PostAdr/street/';
  gPostAdr.street = pData;

when pxPath = '/PostAdr/cty/';
  gPostAdr.cty = pData;

when pxPath = '/PostAdr/state/';
  gPostAdr.state = pData;

when pxPath = '/PostAdr/zip/';
  gPostAdr.zip = pData;

when pxPath = '/PostAdr/phone/';
  gPhnCnt = gPhnCnt + 1;
  gPostAdr.phone(gPhnCnt) = pData;

endsl;
/end-free
P      e

```

```

-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: Using the template engine compose the response and send it to standard out (i.e.
//         RXS_STDOUT).
//
//         rxs3.tpl exists at location /www/myrxs/templates/ which is the default location for
//         templates.
//
//         Update the template with variable data by using the RXS_updVar sub-procedure.
//
//         Finally, write out the last section making sure to specify *on to flush all buffered
//         data.
-----

```

```

P compose          b
D compose          pi
/free

// Compose xml
RXS_initTplEng(RXS_STDOUT: *omit: *omit: *omit: *omit: *on);
RXS_loadTpl('rxs3.tpl');
RXS_wrtSection('HTTP_HEAD');

// Declare the HTTP XML head sent
gSndHdr = *off;

RXS_updVar('status': 'success');
RXS_updVar('message':
  'PostAdr for ' + gPostAdr.firstName + ' ' +
  gPostAdr.lastName + ' has been processed. ');
RXS_wrtSection('output': *on);

/end-free
P          e

```

```

-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: If an error occurs this local sub procedure will be called by the parser.
-----

```

```

P errHandler      B
D errHandler      PI
D pCurLine       10i 0 value
D pCurCol        10i 0 value
D pErrStr         1024a value varying
/free

gError.code = 'RXS3.1';
gError.severity = 100;
gError.pgm = 'RXS3.errHandler';
gError.text =
  'Line:' + %char(pCurLine) +
  ' Column:' + %char(pCurCol) +
  ' ' + pErrStr;

/end-free
P          E

```

```

-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: RXS_stdOutError will compose a simple xml document to send to standard out in
//         response to the request that was made.
-----

```

```

P errRsp          B
D errRsp          PI
/free

RXS_stdOutError('error': gError: gSndHdr);

/end-free
P          E

```

Template: rxs3.tpl

```
/$HTTP_HEAD  
Content-type: text/xml  
Pragma: no-cache  
Expires: Saturday, February 15, 1997 10:10:10 GMT
```

```
/$output  
<response status="%status%/">  
/%message%/</response>
```

Call the RPG Web Service using the included client side web Service Tester

The screenshot shows a window titled "Web Service Tester" with a blue title bar. The interface includes several input fields and buttons:

- URL:**
- SOAP Action:**
- Content-Type:**
- Iterations:** **Time:** 0 seconds
- Buttons:** "Get It" and "Clear Results"

The main content area is divided into three sections:

- Request:** Contains an XML snippet:

```
<PostAdr residential="true">  
  <name title="Mr.">  
    <first>Aaron</first>  
    <last>Bartell</last>  
  </name>  
  <street>123 Center Rd</street>  
  <city>Mankato</city>  
  <state>MN</state>  
  <zip>56001</zip>  
  <phone>123-123-1234</phone>  
  <phone>321-321-4321</phone>  
</PostAdr>
```
- Result:** Contains an XML response:

```
<response status="success">  
PostAdr for Aaron Bartell has been processed.  
</response>
```
- Log:** Contains a single log entry:

```
{Content-Length=[86], Connection=[Keep-Alive], Expires=[Saturday, February 15, 1997 10:10:10 GMT], null=[HTTP/1.1 200 OK], Date=[Fri, 14 Apr 2006 19:11:...
```

The window has a standard Windows-style scrollbar at the bottom.