SDN Contribution

HOWTO: SCRIPTING LANGUAGE SUPPORT FOR SAP SERVICES - PHP

Applies To

SAP NetWeaver; PHP 5; SAPRFC 1.4.1 PHP Extension.

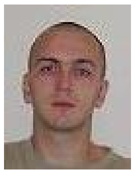
Summary

This article gives an introduction to usage of SAP services in PHP scripts. These concepts are applicable to any scripting language. There are also additional documents concerning Ruby, Python and Perl support. The article does not consider security aspects, since they are subjects of different documents.

Created: 03 May 2006

Version: 1.0.8

Author Bio



Vasil Bachvarov (vasil.bachvarov@sap.com)

has been working for SAP AG since January, 2005. He has developed demo applications and tests for the Composite Applications Framework project, Guided Procedures and does research in the area of scripting language integration with SAP software. Currently considered languages are PHP, Ruby, Python and Perl.

Vasil is a graduate student at the Technical University of Darmstadt, Germany, major Informational and Communicational Technology. He has about 6 years experience in software development.

Table of Contents

Motivation

SAP systems are accessible from the outer world through different interfaces. There are RFC connectors for Java, .NET, the RFC Library for native applications, etc. Another option is accessing SAP enterprise services via web service calls. The connectivity of a scripting language to a SAP system could be achieved by using a special connector, written as an extension to a language's engine (for RFC connectivity), or by using standard web service libraries (for enterprise services connectivity). This document explains the necessary steps to create a SAP-enabled scripting application, using these interfaces and connectors. PHP samples are available within the text.

Writing a PHP Scripting Application, Utilizing RFCs

For this purpose one needs a specific SAP connector for PHP. In the following an example of a PHP application, utilizing RFC is shown.

Assumption: This document assumes that an appropriate PHP scripting environment is already installed and configured.

Example PHP Application with RFC Access

Preparatory Steps

We need to install a PHP extension that will allow us to use the RFC API in the scripts.
An RFC connector for PHP is available at http://saprfc.sourceforge.net and is called **SAPRFC**. Please, follow the installation notes in its documentation in order to install it on your system.

Opening an RFC Connection

To open an RFC connection, pass the connection parameters to the SAPRFC module:

```
$conn = array (
      "ASHOST"   => "ashost1",    # Application host
      "SYSNR"    => "00",         # System number
      "CLIENT"   => "100",
      "USER"     => "jsmith",     # Logon user name
      "PASSWD"   => "drowssap",   # Logon password
      "GWHOST"   => "gwhost1",    # Gateway host
      "R3NAME"   => "ABC",        # R/3 name
      "LANG"     => "EN");        # Language

$rfc = saprfc_open ($conn);
if (! $rfc )
{
      die("RFC connection failed with error:" . saprfc_error());
}
```

You can add other parameters, according to the RFC Library documentation, function RfcOpenEx.

Discovering the Required Function

We have to discover the remote function and get a reference to it before we can make the call. This will automatically retrieve all the necessary parameter definitions of the function.

```
$fce = saprfc_function_discover($rfc, "RFC_FUNCTION1");
if (! $fce )
{
     die("Discovering interface of function module RFC_FUNCTION1 failed");
}
```

Making the Function Call

1. Import the input parameters:

```
saprfc_import ($fce, "PARAM1”, $param1);
saprfc_import ($fce, "PARAM2”, $param2);
```

2. Execute the function:

```
$rc = saprfc_call_and_receive ($fce);
```

3. Check the result for errors:

```
if ($rc != SAPRFC_OK)
{
     if ($rfc == SAPRFC_EXCEPTION )
          echo ("Exception raised: " . saprfc_exception($fce));
     else
          echo ("Call error: " . saprfc_error($fce));
     die();
}
```

4. Export the output parameters:

```
$result = saprfc_export ($fce, "OUTPARAM1");
```

5. Dealing with tables. If there is a table as an output parameter, one could retrieve the information from the table with this code:

```
$return = array();

# Copy the result.
$rows = saprfc_table_rows ($fce, "OUTPUTTABLE1");

for ($i=1; $i<=$rows; $i++)
{
     $MYTABLE = saprfc_table_read ($fce, "OUTPUTTABLE1", $i);
     $return[$i] = $MYTABLE;
}
```

Freeing the Function Reference

```
saprfc_function_free ($fce);
```

Closing the RFC Connection

```
saprfc_close ($rfc);
```

Writing a PHP Application, Consuming SAP Enterprise Services

Enterprise services can be consumed, using the standard web service consumption libraries, available for the scripting environment. In PHP 5 there is built in web service support, available via the SOAP classes.

Example of a PHP Application with Enterprise Service Access

In the next example we'll use the SoapClient class to make the web service call.

Preparatory Steps

First we have to create a SAP enterprise service, which will be consumed, which is out of the scope of this document. Please, refer to the SAP NetWeaver Developer's Guide for more information. Please note, as of release 6.40 and higher all remote enabled functions can be called via this method.

We will use the WSDL description of our enterprise service and the SoapClient class to parse it and create the necessary client stubs for us. Due to some limitations of the currently available web service connectors for scripting languages (missing WSDL import support), we need to modify the SAP WSDL in order to match their constraints and particularly the constraints of the SoapClient class. We will remove the imports statements from the WSDL file and manually insert the imported files on their place.

*Detailed description of this procedure is available in SAP Notes **766953** and **738912**.*

1. Save the WSDL file, located on the NetWeaver application server, and its two imported files (binding and port type WSDL). Access URL is for example http://myserver:50000/MyService/Config1?wsdl&style=document;

2. Open the main WSDL and replace the import statements with the contents of the other two files;

3. Update the name spaces (according to SAP Note 738912).

Now we are ready to use this WSDL in our script.

Creating the SoapClient Object

The proxy object can be created with the following code:

```
$client = new SoapClient( 'path/to/MyService.wsdl');
```

Executing the Web Method

The execution of the method is as simple as calling an automatically generated method stub of the proxy object:

```
try
{
      $res = $client->mymethod(array ('myparam1'=>$myparam1))->Response;
```

```
    }
    catch( SoapFault $e )
    {
          echo $e->faultstring;
    }
```

In case of error an exception is thrown from the method, so we can do graceful error handling.

Other Web Service Consumption Libraries for PHP

There are a number of other web service consumption libraries available. Some of them are: NuSOAP (http://dietrich.ganx4.com/nusoap/), PEAR::Soap (http://pear.php.net/package/SOAP). One should however consider their particular limitations and characteristics, when consuming SAP enterprise services.

Conclusion

SAP service consumption in PHP scripts can be implemented by calling enterprise services or RFCs. Enterprise services are compliant with the standard web services and therefore can be executed using the built in PHP libraries for web service calls. RFCs are available in PHP scripts thanks to the SAPRFC extension module, which exposes the RFC library functions to the PHP application.

Related Content

Scripting Languages Support for SAP Services

I. SAP Developer Network (http://sdn.sap.com).

RFC Consumption in PHP

II. SAPRFC Connection Module for PHP (http://saprfc.sourceforge.net).

Web Services Consumption in PHP

III. Solving the WSDL Import Problem for SAP Enterprise Services (SAP Notes **766953** and **738912**).

IV. PHP SOAP Functions (http://de3.php.net/manual/en/ref.soap.php).

V. NuSOAP – Web Service Consumption Library (http://dietrich.ganx4.com/nusoap/).

VI. PEAR::Soap – Web Service Consumption Library (http://pear.php.net/package/SOAP).

VII. Zend SOAP Information (http://www.zend.com/php5/articles/php5-SOAP.php).

NetWeaver Development

VIII. SAP NetWeaver Developer's Guide (https://www.sdn.sap.com/irj/sdn/developersguide).

Disclaimer and Liability Notice