WSDL2RPG – FAQ

FAQ Useful Procedures

Status of this Document

Date:	08.02.2012
Version:	0.2 (under construction)

Question

Actually there is no question “Useful Procedures”. This document is intended to explain the various procedures of the web service stub module. Maybe you can take advantage from it.

Notation

“web service stub module”	The module that contains the procedures which are common to all operations of the web service.
“web service operation module”	The module that is specific to a particular web service operation. A complete web service stub consists of one or more operation modules and one stub module.

“Endpoint” Procedures

“Endpoint” is the name of the URL of the web service. Do not confuse it with the URL of the WSDL file. The URL of the WSDL file is used to download the WSDL file whereas the endpoint URL is the URL that is used to call a web service operation.

Endpoint_set()

Specifies the endpoint of the web service. This procedure can be used to change the endpoint to a different URL. For example you may call this procedure if you want to switch from a production to a test web service with a completely different URL.

See also: `Endpoint_setHost()`
`Endpoint_setPort()`
`Port_initialize()`

Endpoint_get()

Returns the complete URL of the web service endpoint. In fact this URL should match the URL given in the WSDL file at (sample):

```
<service name="ZipCode">
  <port name="ZipCodeSoap" binding="s0:ZipCodeSoap">
    <soap:address location="http://www.ripedevelopment.com/webservices/ZipCode.asmx" />
  </port>
</service>
```

The complete WSDL file is available at:

http://www.ripedevelopment.com/webservices/ZipCode.asmx?wsdl

This procedure is called from the web service operation module right before sending the request message to the server.

Endpoint_setHost()

Specifies the host of the web service endpoint. Use this procedure to only change the host name of the endpoint of the web service. You may use this procedure to switch from a production to a test web service, for example:

```
http://prod.tools400.dyndns.org:88/axis2/services/DownloadAttachmentService

http://test.tools400.dyndns.org:88/axis2/services/DownloadAttachmentService
```

Note: Both URLs are fictive addresses and do not exist.

See also: `Port_initialize()`

Endpoint_getHost()

Returns the host name of the current endpoint URL.

Endpoint_setPort()

Specifies the port number of the web service endpoint. Use this procedure to change only the port number of the endpoint URL of the web service. You may use this procedure to switch from a production to a test web service, for example:

```
http://tools400.dyndns.org:80/axis2/services/DownloadAttachmentService

http://tools400.dyndns.org:8080/axis2/services/DownloadAttachmentService
```

Note: Both URLs are fictive addresses and do not exist.

See also: `Port_initialize()`

Endpoint_getPort()

Returns the port number of the current endpoint URL.

“Port” Procedures

The “Port” procedures control how to connect to the web service. Most of these procedures are closely tied to HTTPAPI.

Port_initialize()

Used to initialize the web service port. This operation is the very first procedure that is called when you call a web service operation. By default this procedure registers procedure `Port_supplyLoginData()` which sends error message “Failed to get login credentials” (USR0048) in case a web service requires authentication. Either register your own callback procedure or modify `Port_supplyLoginData()` to fit your needs. Feel free to use window `WSDL2R42_getLoginData()` if you want the user to enter name and password.

You may also use this procedure to get environment (Test, QA, Prod) specific parameters from a physical file, such as:

- Debug mode
- Web Service Endpoint

Just use the value of `Port_getName()` or `Port_getUuid()` as the key to read these settings.

See also:	`Port_registerLoginCallback()`
	`Port_supplyLoginData()`
	`Port_setPostCcsid()`
	`Port_setTimeout()`
	`Port_setUserAgent()`
	`Endpoint_set()`
	`Endpoint_setHost()`
	`Endpoint_setPort()`

Port_setHttpDebug()

Used to turn the HTTPAPI debug log on or off. You should use this procedure to turn the debug log on if you encounter problems with the web service. The debug log is nothing else but a trace log of the communication between the client and the server. It is essential to have the debug log when debugging a web service stub.

See also: `Port_initialize()`

Port_getHttpDebug()

Returns the current status of the debug log. It is used by the web service operation module to get the actual debug settings. Feel free to modify this procedure for example in case you always want to write a debug log or to produce a unique log file name for each web service call.

Port_setPostCcsid()

Sets the remote CCSID of post data. The default value is UTF-8 which is commonly used by web services.

See also: `Port_initialize()`

Port_getPostCcsid()

Returns the CCSID of the post data. The web service operation module uses it to forward the CCSID to procedure `HTTP_SetCCSIDs()`. Feel free to modify this procedure if UTF-8 does not match your needs.

Port_setHttpProxy()

Specifies the name and port number of the proxy server. A lot of companies use proxy servers to connect to the Internet for security reasons. If your network also requires using a proxy server, you need to call this procedure to set the name and port number of the proxy server for HTTPAPI.

See also: `Port_initialize()`

Port_getHttpProxy()

Returns the current proxy server settings. The web service operation module calls this procedure to forward the proxy settings to HTTPAPI. Feel free to modify this procedure to match your needs.

Port_supplyLoginData()

Supplies login credentials for web services requiring authentication. The sample implementation sends escape message “Failed to get login credentials” (USR0048). You may change this procedure to fit your needs.

See also: `Port_initialize()`
`Port_registerLoginCallback()`

Port_registerLoginCallback()

Registers a callback procedure that is called when the web service requires authentication. Use this procedure to register a callback procedure that supplies login credentials for web services that require authentication. Feel free to use window `WSDL2R42_getLoginData()` if you want the user to enter name and password or develop your own callback procedure that e.g. may read the credentials from a physical file.

See also: `Port_supplyLoginData()`

Port_hasLoginCallback()

Returns `*ON` if a login callback procedure has been registered, else `*OFF`. This procedure is internally used by the web service operation module to determine whether or not login credentials are available.

Port_login

Performs the login procedure when connecting to the server. This procedure is called from the web service operation module whenever the web service requires authentication and a procedure to supply login credentials has been registered.

Usually it is not necessary to change anything here.

Port_setTimeout()

Specifies the timeout in seconds. This timeout value is used when connection to the server or waiting for response from the server.

See also: `Port_initialize()`

Port_getTimeout()

Returns the current timeout value in seconds. This procedure is used by the web service operation module to retrieve and forward the timeout value to HTTPAPI.

Port_setUserAgent()

Specifies the user-agent value. The user-agent value is used when connecting to the server. It specifies the name and version number of the client. Usually the server uses this value for statistics and sometimes to restrict pages to certain clients, e.g. “Mozilla”.

Usually it is not necessary to change anything here.

See also: `Port_initialize()`

Port_getUserAgent()

Returns the current user-agent value. This procedure is used by the web service operation module to retrieve and forward the user-agent value to HTTPAPI.

Port_isError()

Returns `*ON` if the web service call ended in error, else `*OFF`. This procedure should be used to check for errors after having called the web service operation. The generated sample programs demonstrate how to properly use it.

Errors can occur on different levels of the application. For example there might be connection errors, errors on the remote server while processing your request or XML parser errors when the received response message is parsed. The web service stub module provides procedures for retrieving error information for each of these level:

- HTTP errors
- SOAP errors
- XML parser errors

See also:
```
HttpError_getCode()
HttpError_getText()
SoapError_getCode()
SoapError_getText()
XmlError_getCode()
XMLError_getText()
```

`HttpError_getCode()`

Returns the last HTTP error code such as “Host not found” (`HTTP_HOSTNF`) or “Timeout when connecting to server” (`HTTP_CNNTIMO`).

See also: `HttpError_getText()`

`HttpError_getText()`

Returns the last HTTP error message such as “Host not found” or “Timeout when connecting to server”.

See also: `HttpError_getCode()`

`SoapError_getCode()`

Returns the last SOAP error code such as “VersionMismatch” or “MustUnderstand”. Refer to chapter “4.4.1 SOAP Fault Codes” of document “Simple Object Access Protocol (SOAP) 1.1” for a description of the SOAP Fault Codes.

The server returns a SOAP error for example in case it received the message but there is something wrong with it.

See also: `SoapError_getText()`

`SoapError_getText()`

Returns the last SOAP error message.

See also: `SoapError_getCode()`

`XmlError_getCode()`

Returns the last XML error code such as “mismatched tag” (`XML_ERROR_TAG_MISMATCH`) or “unknown encoding” (`XML_ERROR_UNKNOWN_ENCODING`). The error codes and the error descriptions are defined in `EXPAT.H` and `XMLPARSE.C` of file `EXPAT` in library `LIBHTTP`.

Error code -1009 is set by WSDL2RPG in case of an unexpected but monitored error.

See also: `XmlError_getText()`

`XmlError_getText()`

Returns the last XML error code such as “mismatched tag” or “unknown encoding”. The error codes and the error descriptions are defined in EXPAT.H and XMLPARSE.C of file EXPAT in library LIBHTTP.

See also: `XmlError_getCode()`

Internally Used Procedures

`Port_getName()`

Returns the name of the web service port. Frankly this procedure does not control anything but is internally used to produce error message “Unexpected HTML data received from web service.” (USR0030).

The port name may be used as a key for reading parameter values from a physical file.

See also: `Port_initialize()`

`Port_getUuid()`

Returns the Universal Unique Identifier (UUID) of this web service port. It is internally used to associate dynamic arrays to a particular web service port. It comes into play if you generate the web service port with parameter DIM set to *NOMAX.

The UUID may be used as a key for reading parameter values from a physical file.

See also: `Port_initialize()`

`Port_clearErrors()`

Removes all error information of the previous call of the web service. The web service operation module uses this procedure to clear previous errors before it calls the web service.

Do not change anything here unless you exactly know what you are doing.

`Port_setErrors()`

Sets the current error value. This procedure is used by the web service operation module to set the global error information value.

You should never call this procedure on your own or change anything with it unless you exactly know what you are doing.

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

thomas.raddatz@tools400.de