XML-RPC Libraries

Your programming language may require additional libraries to use XML-RPC client functionality. The following sections provide examples of XML-RPC libraries for common programming languages.

XML-RPC Data Type Compatibility

When an XML-RPC request is sent to the ABC API server, the server uses the method signature to match the request to the corresponding method. The method signature depends on the data types of the passed parameters.

Loosely typed languages require type casting to ensure a method signature match. This functionality is typically provided by XML-RPC libraries. If it is not, define the XML-RPC data types per the XML-RPC specification to ensure compatibility with the ABC API.

Your XML-RPC library may require additional instructions for handling data returned by the ABC API Server. XML-RPC libraries typically provide functionality for handling special data formats that may be incompatible with the language defaults. Because the XML-RPC library that handles data parsing needs instructions for additional formats, any custom format handling instructions are attached to the XML-RPC client instance.

Date formatting is one example of a data format that typically requires additional handling instructions. If your XML-RPC library uses the default date formats for your programming language, you may need to specify additional date format instructions to the XML-RPC client instance. The ABC API uses datetimestamps in the following ISO8601 extended format:

```
yyyy-MM-ddThh:mm:ssZ
```

Examples by Language

Perl

Search a package repository for an XML-RPC client module that works with your version of Perl. The examples in this document use the `RPC::XML::Client` module available via a Perl package repository such as CPAN[1] or ActiveState's PPM[2].

```
use RPC::XML::Client;
```

XML-RPC Typecasting

Perl is a loosely typed language. Cast parameters sent to the XML-RPC API to the appropriate XML-RPC type per the method definition. Use the type casting functions from your XML-RPC module.

The ABC API treats some numerical values, like phone numbers, as strings. Cast these values to string format.

```
RPC::XML::string->new("8005551212")
```

The ABC API uses the ISO8601 datetime format. Cast these values to an ISO8601 datetime format.

```
RPC::XML::datetime_iso8601->new('20070917T000000Z')
```

The XML-RPC `struct` corresponds with Perl's hash. For methods in the ABC API that take a `struct` as a multi-value parameter, cast a hash containing these parameters to an XML-RPC `struct`.

```
RPC::XML::struct->new(%search)
```

For more information about casting Perl data types to XML-RPC data types, see your XML-RPC module documentation.

[1] http://www.cpan.org
[2] http://www.activestate.com/perl

PHP

Search a PHP library repository such as Pear[3], Pecl[4], or Sourceforge[5] for an XML-RPC client package. The examples in this document use the PHPXMLRPC package available on Sourceforge[6].

Download the `xmlrpc.inc` file to a directory that is accessible by the web service and provide the full path in the include statement. The below statement indicates the `xmlrpc.inc` file is in the same location as the PHP script.

```
include ('xmlrpc.inc');
```

Java

The examples in this document use the Apache Foundation's XML-RPC Library[7]. Update your `CLASSPATH` or add the XML-RPC libraries you've installed to your IDE and to your project.

Import the XML-RPC client packages.

```
import org.apache.xmlrpc.client.XmlRpcClient;
import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;
```

If you are using the Apache Foundation's XML-RPC Library you may need to import additional libraries. For more information, see the documentation for the Apache Foundation's XML-RPC Library.[8]

Datetime ISO8601 Compatibility

The ABC API returns date and time values in the ISO8601 extended format. Your Java application may need extra instructions for handling this format.

The Apache Foundation's XML-RPC Library provides the TypeFactory class for creating custom data types. They have an example on their website for ISO8601 date handling[9] (under Custom data types), which we have modified for our date format in the example below.

Modify the example by editing the `newFormat()` method so the returned `SimpleDateFormat` matches the extended ISO8601 format returned by the ABC API.

```
private DateFormat newFormat() {
    new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
}
```

A working example is available for download with the Java example files.

Alternatively, the Joda-time[10] library offers more sophisticated date handling than Java's date libraries. Choose the library that best fits your development needs.

---

[3] http://pear.php.net
[4] http://pecl.php.net
[5] http://www.sourceforge.net
[6] http://phpxmlrpc.sourceforge.net
[7] http://ws.apache.org/xmlrpc
[8] http://ws.apache.org/xmlrpc/apidocs/index.html
[9] http://ws.apache.org/xmlrpc/advanced.html
[10] http://joda-time.sourceforge.net

.NET

The examples in this document use the Cook Computing XML-RPC Library.[11] Install the assemblies and add the corresponding library reference to your .NET project.

Define the XML-RPC client library.

```
using CookComputing.XmlRpc;
```

Datetime ISO8601 Compatibility

The ABC API returns date and time values in ISO8601 extended format. Your .NET application may need extra instructions for handling this format.

The Cook Computing XML-RPC Library defines an enumerated list of properties called `XmlRpcNonStandard` in the `XmlRpcClientProtocol` class. Set the `AllowNonStandardDateTime` flag for the client instance in your application.

```
IABC client = (IABC)XmlRpcProxyGen.Create(typeof(IABC));
client.NonStandard = (XmlRpcNonStandard.AllowNonStandardDateTime);
```

For more information, see the documentation for the Cook Computing XML-RPC Library.[12]

[11] http://www.xml-rpc.net
[12] http://www.xml-rpc.net/xmlrpcnetfaq.html#2.22