



Using SOAP and Web Services with Caché

Version 5.2
01 September 2006

Using SOAP and Web Services with Caché
Caché Version 5.2 01 September 2006
Copyright © 2006 InterSystems Corporation.
All rights reserved.

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.



The Caché product and its logos are registered trademarks of InterSystems Corporation.



The Ensemble product and its logos are registered trademarks of InterSystems Corporation.



The InterSystems name and logo are trademarks of InterSystems Corporation.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

Caché, InterSystems Caché, Caché SQL, Caché ObjectScript, Caché Object, Ensemble, InterSystems Ensemble, Ensemble Object, and Ensemble Production are trademarks of InterSystems Corporation. All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support

Tel: +1 617 621-0700
Fax: +1 617 374-9391
Email: support@InterSystems.com

Table of Contents

1 Introduction	1
1.1 Caché SOAP Server Architecture	1
1.2 Caché SOAP Client Architecture	3
2 Creating a Web Service	5
2.1 Defining a Web Service	5
2.1.1 Web Service Parameters	6
2.1.2 Creating a New Web Service Using Studio	7
2.2 Defining Web Methods	7
2.2.1 Literal-valued Method Arguments	8
2.2.2 Object-valued Method Arguments	8
2.3 Using a Web Service	9
2.3.1 Testing a Web Service from a Browser	9
2.3.2 Using a Web Service from Java	11
2.3.3 Using a Web Service from .net	11

List of Tables

Web Service Parameters	7
------------------------------	---

1

Introduction

Caché provides full support for SOAP (Simple Object Access Protocol). This support is easy-to-use, efficient, and fully compatible with the SOAP specification. This support is built into Caché and does not require any complex middleware or operating system extensions. It is available on every platform supported by Caché.

SOAP is an HTTP-based protocol that allows applications to invoke remote procedure calls across the internet. SOAP uses XML to encode requests and their subsequent responses.

Using Caché SOAP, you can:

- Define and publish Web Services—a collection of related methods that client applications invoke using the SOAP protocol. Such methods can be discovered and invoked by other SOAP-aware applications. Caché runs SOAP methods directly within the database—not within external middleware—making the execution of such methods highly efficient.
- Provide a way for your applications to interoperate with other applications using the standard SOAP protocol.
- Develop highly efficient Web Services (based on new development or perhaps utilizing existing application code) that can be deployed on any platform supported by Caché—you are not limited to a specific vendor's operating system or platform.

1.1 Caché SOAP Server Architecture

The Caché SOAP Server works as follows:

- For each Web Service (a set of related SOAP methods) you wish to implement, you create a new Caché class definition that extends (is derived from) the %SOAP.WebService class provided with the Caché library.
- Within this class definition you define one or more class methods that correspond to the methods of your Web Service. Each of these can be specified as being a “WebMethod” by adding the “WebMethod” keyword to its definition.

You can also define Web Methods that return arrays of “objects” by defining a class query (basically a built-in result set based either on an SQL statement or user-defined code) and adding the “WebMethod” keyword to its definition.

Note that SOAP is a stateless protocol and does not support invoking instance methods. This is why only class methods can be marked as Web Methods. Within a web method, however, you are free to execute all kinds of code including creating object instances, Caché database access, remote database access via the Caché SQL Gateway, etc.

- When you compile a Web Service class, the Caché Class Compiler automatically assembles catalog information describing the contents of the SOAP service and builds a SOAP interface for each Web Method. The SOAP interface for a Web Method is a generated class that performs the work of converting the SOAP request into a specific call to the Web Method using Caché’s XML-to-object conversion technology.
- Caché automatically publishes a WSDL (Web Services Description Language) document for each Web Service. The WSDL document is an XML document providing a list of available methods, their signatures, and details on how they can be invoked from a SOAP client. The WSDL document is published via a Web (HTTP) Server using CSP (Caché Server Pages).

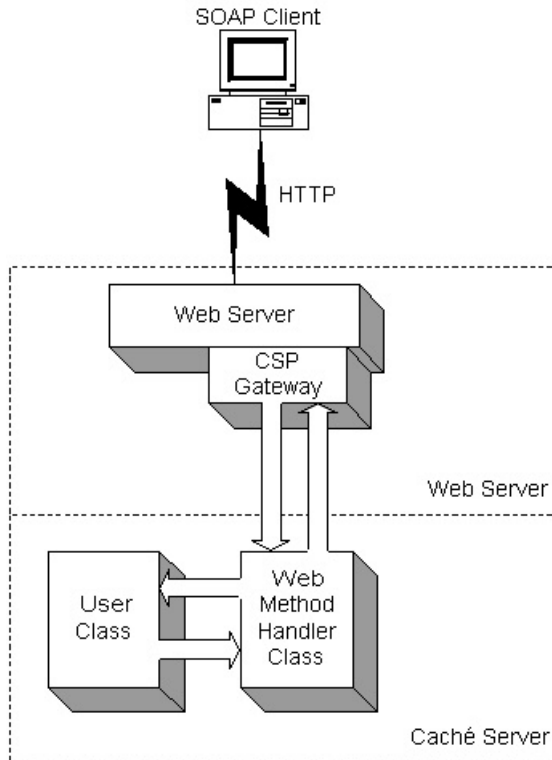
The WSDL document is served *dynamically* and will automatically reflect any changes you make to the interface of your Web Service class.

- A SOAP client “discovers” an available web service by requesting the WSDL document from a Web Server which, in turn, requests it from the Caché server. Using this information in the WSDL document, the SOAP client invokes a specific method by constructing an XML message (referred to as an XML envelope) and posting it (via HTTP) to the SOAP server as specified in the WSDL document.
- The Caché SOAP server receives the SOAP request via the Caché (CSP) HTTP Gateway. The server unpacks the message, validates it, and invokes the specified web (class) method. Before invoking the web method, the Caché SOAP server converts all input parameters to their appropriate Caché representation. In the case of complex types this

include creating instances of objects that represent the complex type and using them as input arguments for the web method.

- The web method executes its code and returns a response writing out an XML response to the current output device. This response can be a simple literal value (e.g., a string); an XML-encoded object, or set of objects; or, the results of a query (such as an SQL query).

The following diagram illustrates how a SOAP message is executed:



1.2 Caché SOAP Client Architecture

Caché provides the ability to create a SOAP client: a class containing methods which, when called, invoke a Web Service method on another system using the SOAP protocol.

The Caché SOAP Client works as follows:

- For each Web Service (a set of related SOAP methods) you wish to invoke, you create a new Caché class definition that extends (is derived from) the %SOAP.WebClient class provided with the Caché library.
- The SOAP client class contains one or more class methods that correspond to the methods of the Web Service it wishes to connect to. Each of these client methods is specified as being a “WebMethod” by adding the “WebMethod” keyword to its definition.

Note that SOAP is a stateless protocol and does not support invoking instance methods. This is why only class methods can be marked as Web Methods. Within a web method, however, you are free to execute all kinds of code including creating object instances, Caché database access, remote database access via the Caché SQL Gateway, etc.

- When you compile a SOAP Client class, the Caché Class Compiler automatically assembles catalog information describing the contents of the SOAP service and builds a SOAP client interface for each Web Method. The SOAP client interface for a Web Method is a generated class that performs the work of converting the SOAP request into a specific call to the Web Method using Caché’s XML-to-object conversion technology.
- A SOAP client “discovers” an available web service by requesting the WSDL document from a Web Server which, in turn, requests it from the Caché server. Using this information in the WSDL document, the SOAP client invokes a specific method by constructing an XML message (referred to as an XML envelope) and posting it (via HTTP) to the SOAP server as specified in the WSDL document.
- A SOAP server receives the SOAP request, unpacks the message, validates it, and invokes the specified operation.

2

Creating a Web Service

In this chapter we will describe how to define a Web Service within Caché and how to use this service from a SOAP client.

2.1 Defining a Web Service

Within Caché, every Web Service is defined by a subclass of the `%SOAP.WebService` class. The `%SOAP.WebService` base class provides all the functionality required to make one or more methods callable via the SOAP protocol. In addition, this class automates the management of SOAP-related “bookkeeping”, such as maintaining a WSDL document that describes a service.

Here is the definition of a simple Caché Web Service. In this case, the `SOAP.Demo` class that is provided within the Caché `SAMPLES` namespace:

Creating a Web Service

```
// A simple demonstration Web Service
Class SOAP.Demo Extends %SOAP.WebService
{
  // This is the name of our service.
  Parameter SERVICENAME = "SOAPDemo";

  // Change this to your own namespace when you deploy a service.
  Parameter NAMESPACE = "http://tempuri.org";

  // Returns a corporate mission statement.
  ClassMethod Mission() As %String [WebMethod]
  {
    Quit ##class(%PopulateUtils).Mission()
  }

  // Returns the city name for the given U.S. ZIP Code.
  ClassMethod LookupCity(zip as %String) As %String [WebMethod]
  {
    New SQLCODE, city
    Set city = ""

    // Use embedded SQL to perform the query
    &sql(SELECT City INTO :city
        FROM Sample.USZipCode
        WHERE ZipCode = :zip )

    Quit city
  }
}
```

This class defines a Web Service names “SOAPDemo” . The Web Service has two methods: **Mission**, which returns a string containing a corporate mission statement (implemented using the %PopulateUtils data population class); and **LookupCity**, which looks up a city name in a U.S. ZIP code database (the Caché SAMPLES database includes a database of cities in the northeastern portion of the U.S. for demonstration purposes).

To try out this Web Service, use one of the techniques described below in [Using a Web Service](#).

2.1.1 Web Service Parameters

The %SOAP.WebService class defines a set of parameters that specify the characteristics of a Web Service. These parameters include:

Web Service Parameters

Parameter	Description
NAMESPACE	The URI used to define a namespace for your web service. A namespace is used to ensure that your service, and its contents, do not conflict with another service. By default this is set to "http://tempuri.org" —a URI used by SOAP developers as a temporary namespace during development. At deployment, you should change this.
SERVICENAME	This is the name of the Web Service. This name should be a valid identifier: i.e, start with a letter and contain no non-alphanumeric characters.

2.1.2 Creating a New Web Service Using Studio

Creating a new Web Service class with Caché Studio is simple:

1. Within Studio, invoke the **New** command within the **File** menu.
2. This displays the New Document Dialog. Choose “New Web Service” and press the **OK** button.
3. This displays the Web Service Wizard. Fill in values for the class name as well as the Web Service parameters (described in [Web Service Parameters](#)) and press the **OK** button.

At this point you will have a new Web Service class such as:

```
Class MyApp.StockService Extends %SOAP.WebService
{
Parameter SERVICENAME = "MyStockService";
Parameter NAMESPACE = "http://tempuri.org";
}
```

From here you can add one or web methods to your class. This is described in the following section.

2.2 Defining Web Methods

A web method is simply a class method of a Caché class that is marked as being a `WebMethod`. For example:

```
Class MyApp.StockService Extends %SOAP.WebService
{
Parameter SERVICENAME = "MyStockService";
Parameter NAMESPACE = "http://tempuri.org";

/// This method returns tomorrow's price for the requested stock
ClassMethod Forecast(StockName As %String) As %Integer [WebMethod]
{
    // apply patented, nonlinear, heuristic to find new price
    Set price = $Random(1000)
    Quit price
}
}
```

When this class is compiled, it will publish a Web Service called “MyStockService” containing a single method called “Forecast”.

A Web Method can contain any implementation code you like including database access, embedded SQL queries, object access, etc. The only restriction is that any non-literal input parameters or return values must be represented as XML-enabled classes, that is classes that include %XML.Adaptor as a super class.

2.2.1 Literal-valued Method Arguments

Simple, literal method parameters and return types are represented using Caché data type classes. For example, a Web Method that takes 2 integers as input values and returns a string would be defined as:

```
ClassMethod MyMethod(p1 As %Integer,p2 As %Integer) As %String [WebMethod]
{
    Quit "A string"
}
```

2.2.2 Object-valued Method Arguments

Complex, that is composite, method arguments and return types are represented using objects. Specifically, non-data-type, XML-enabled Caché classes.

For example, a Web Method that takes 2 complex numbers (with real and imaginary components) as input values and returns a complex number would be defined as:

```
ClassMethod Add(a As ComplexNumber,b As ComplexNumber)
    As ComplexNumber [WebMethod]
{
    Set sum = ##class(ComplexNumber).%New()
    Set sum.Real = a.Real + b.Real
    Set sum.Imaginary = a.Imaginary + b.Imaginary

    Quit sum
}
```

In this case `ComplexNumber` is defined as follows:

```
/// A complex number
Class MyApp.ComplexNumber Extends (%RegisteredObject,%XML.Adaptor)
{
Property Real As %Float;
Property Imaginary As %Float;
}
```

2.3 Using a Web Service

Once you have defined a Web Service in Caché, you can try it out in a variety of ways. Some of these are described in this section.

Within this section, we will use the sample Web Service class included with Caché.

2.3.1 Testing a Web Service from a Browser

Every Caché Web Service automatically provides a simple HTTP-based interface that allows you to discover and test your web services using a web browser. Internally this works as follows: every Web Service class is derived from the `%SOAP.WebService` class which, in turn, is derived from the `%CSP.Page` class. Because of this, every Web Service class can respond to HTTP requests (via CSP). The `%SOAP.WebService` class implements methods that respond to HTTP events to do the following:

- Publish a human-readable Catalog page (using HTML) describing the Web Service and its methods.
- Publish the WSDL document for the Web Service as an XML document.
- Publish an HTML-based test page for each Web Service method allowing you to test each method interactively.

2.3.1.1 The Web Service Catalog Page

To display the Web Service Catalog page for a specific Web Service within a browser, request the URL that corresponds to your Web Service class (using the CSP `.cls` file extension). For example, if your web service class is called `MyApp.StockService` and is defined within the `USER` namespace of your Caché server, you would use the following URL:

```
http://localhost/csp/user/MyApp.StockService.cls
```

This will display the Catalog page for the `MyApp.StockService` class as an HTML page.

To view the Catalog page for the sample Web Service included with Caché, visit the following URL from a browser: <http://localhost/csp/samples/SOAP.Demo.cls>

2.3.1.2 The WSDL Document

To display the WSDL document for a specific Web Service, request the URL that corresponds to your Web Service class (using the CSP .cls file extension) and include “WSDL” in the URL parameter list. For example, if your web service class is called MyApp.StockService and is defined within the USER namespace of your Caché server, you would use the following URL:

```
http://localhost/csp/user/MyApp.StockService.cls?WSDL
```

This will display the WSDL document for the MyApp.StockService class as an XML document.

To view the WSDL document for the sample Web Service included with Caché, visit the following URL from a browser: <http://localhost/csp/samples/SOAP.Demo.cls>

You can also create a static XML file containing the WSDL document for your web service by using the Web Service class' **FileWSDL** method (inherited from the %SOAP.WebService class). For example:

```
Do ##class(MyApp.StockService).FileWSDL("/wsdl/StockService.xml")
```

2.3.1.3 The Web Method Test Page

The Web Method Test Page is a web page that, for a specific web method, displays a form in which you can enter values for the web method's input parameters. You can then invoke the web method by pressing the **Submit** button. The data returned from the Web Method is then displayed, as XML, within the browser.

To display the Web Method Test Page for a specific Web Method within a browser, request the URL that corresponds to your Web Service class (using the CSP .cls file extension) and follow it with a “/” and the method name. You can add any method arguments as URL parameters following the method name.

For example, if your method is called **Forecast** and your web service class is called MyApp.StockService and is defined within the USER namespace of your Caché server, you would use the following URL:

```
http://localhost/csp/user/MyApp.StockService.cls/Forecast
```

This will display the Web Method Test Page page for the **Forecast** method of the MyApp.StockService class.

To view the Web Method test page for the sample Web Service included with Caché, visit the following URL from a browser: <http://localhost/csp/samples/SOAP.Demo.cls> and click on one of the links to the various web method test pages.

2.3.2 Using a Web Service from Java

You can invoke Web Service methods from Java by using the SOAP client classes available for Java.

2.3.3 Using a Web Service from .net

You can invoke Web Service methods from .net by using the SOAP client classes included with the .net framework. As SOAP is an open, standard protocol it makes no difference to a SOAP client where, how, or on what platform a Web Service is implemented.

For example, suppose we want to use the sample “SOAPDemo” Web Service (included with the SAMPLES database) from Visual Basic .net.

First, add a Web Reference to your Visual Basic project. This gives the SOAP client the information it needs (via a WSDL document) to interact with a given Web Service. You can add a Web Reference within Visual Studio .net by invoking the **Add Web Reference** command within the **Project** menu. This will prompt you for the address (URL) of the WSDL document for a Web Service. In this case, enter the following URL in the **Address** field:

```
http://localhost/csp/samples/SOAP.Demo.cls?WSDL
```

This will serve up the WSDL document for the SOAPDemo service.

Next, you use the Web Service in your client application via a SOAP client interface object. In this case the SOAP client object is an instance of SOAPDemo class. The following Visual Basic code creates an instance of the SOAP client interface, *tMyService*; invokes the methods of the “SOAPDemo” Web Service; and displays their results in a message box:

```
'tMyService is an instance of SOAPDemo
'Show a mission statement
MsgBox.Show(tMyService.Mission())

'Loopkup up a city for a zipcode
MsgBox.Show(tMyService.LookupCity("02139"))
```

In a similar fashion, you can do the same from another .net language.

