HR/Benefits Extend Implementation Guide

Version: 2.0
3/16/2010
ADP’s Trademarks
The ADP Logo is a registered trademark of ADP of North America. HR Benefits Solution, AutoPay and Pay eXpert are registered trademarks of ADP, Inc.

Third-Party Trademarks
Adobe, Acrobat, and Acrobat Reader are registered trademarks of Adobe Systems Incorporated in the United States and/or other countries. Citrix GoToAssist is a trademark or registered trademark of Citrix Online, LLC. Macromedia is a registered trademark of Macromedia, Inc. Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation in the United States and/or other countries. Seagate is a registered trademark of Seagate Technology, Inc. WinZip is a registered trademark of WinZip Computing, Inc. All other trademarks are the property of their respective owners.

Copyright © 2010 ADP, Inc. All rights reserved.

U.S. Government Restricted Rights. The Software and written materials accompanying the Software are “commercial computer software” or “commercial computer software documentation.” Absent a written agreement to the contrary, the Government’s rights with respect to such Software or materials are limited by the terms of the applicable product/service license agreement, pursuant to FAR’ 12.212(a) and/or DFARS’ 227.7202-1(a), as applicable. The information contained herein constitutes proprietary and confidential information of ADP. It must not be copied, transmitted, or distributed in any form or by any means, electronic, mechanical, or other, including photocopy, recording, or any information storage and retrieval system, without the express written permission of ADP. ADP provide this publication “as is” without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. ADP is not responsible for any technical inaccuracies or typographical errors which may be contained in this publication. Changes are periodically made to the information herein, and such changes will be incorporated in new editions of this publication. ADP may make improvements and/or changes in the product and/or the programs described in this publication at any time without notice.
Overview ............................................................................................................................ 4
Introduction .......................................................................................................................... 4
Audience............................................................................................................................... 4
Document Status .................................................................................................................. 4
Conventions ......................................................................................................................... 4

Building a Simple Web Service Call Using .NET ............................................................. 5
Tools ..................................................................................................................................... 5
Set-Up .................................................................................................................................. 5
  Create a new project .......................................................................................................... 5
  Add a reference to WSE 3.0 ............................................................................................... 5
  Add a web reference using WSDL ..................................................................................... 7
Building a simple web service request ............................................................................. 10
  Initializing the web proxy client object ........................................................................... 10
  Authentication ................................................................................................................ 11
  Building the request payload ......................................................................................... 12
  Parsing the response payload ......................................................................................... 13
  .NET Sample Code ........................................................................................................ 14

Applying Filters to Get Services ..................................................................................... 16

Applying Filters to Get Services ..................................................................................... 16
  Common Filters ................................................................................................................ 16
  Employee Key Filter ........................................................................................................ 16
  Company Filter ............................................................................................................... 16
  Last Name Filter .............................................................................................................. 17
  Combination Filters ........................................................................................................ 17
  GetEmployeeStatus Example ....................................................................................... 17
  GetEmployeeCorpGroup Example ................................................................................ 18

Appendix .......................................................................................................................... 19
  Related Documents........................................................................................................ 19
  Example Payloads .......................................................................................................... 19
  GetEmployeeCorpGroup Request ................................................................................ 19
  GetEmployeeCorpGroup Response ............................................................................... 20
  HireEmployee Request .................................................................................................. 21
  HireEmployee Response ............................................................................................... 23
Overview

Introduction
HR/Benefits Extend is a collection of flexible web services that enable provisioning of data, consumption of data and execution of processes within the HR/Benefits Solution. This separates the presentation layer from the application layer so that organizations can integrate with any other system or entity across the network regardless of programming language, platform, device, or web browser.

Audience
This tutorial is intended for software developers who are looking to implement web services using HR/Benefits Extend.

Document Status
Version 2.0 – 3/16/2010 – Update to new security standards requiring mutual SSL and signed requests.

Conventions
Code snippets will appear using a mono-spaced font and a blue color scheme.

Example
for (int i = 0; i < 10; i++)
{
    System.Console.Out.WriteLine(i);
}

Useful tips and best practices will appear using the following bullet style.

Example
- Work closely with your HR administrator to accurately model your company’s business processes.
Building a Simple Web Service Call Using .NET

Tools
The following tool sets are used during this tutorial. Some steps may need to be modified when working with versions other than what is listed below.

- Microsoft Visual C# 2008 Express Edition
- Microsoft .NET Framework 3.5
- Web Services Enhancements (WSE) 3.0 for Microsoft .NET
- XML Editor (Optional, but recommended)

Set-Up

Create a new project
To simplify the code, we will be using creating a console application using the "Console Application" template in the New Project wizard. To remain consistent throughout the tutorial, please name this project "TestApp".

Add a reference to WSE 3.0
In order to instantiate a UserToken conforming to the WS-I basic security profile, we will need to add a reference to the WSE 3.0 package. This can be done by using the main menu items Project->Add Reference.
From the "Add Reference" pop-up, stay on the .NET tab and select the `Microsoft.Web.Services3` package. Click OK.
Add a web reference using WSDL

Adding a web reference allows the .NET framework to auto-generate most of the functionality necessary to make a web service call. This can be done using the main menu items Project->Add Service Reference.
This should bring up the “Add Web Reference” pop-up, which is shown below.

Click the “Advanced” button in the lower left hand corner. This should bring up the Service Reference Settings pop-up.
Click on the "Add Web Reference" button in the Compatibility section.
In the URL bar, enter the location of the WSDL file that you were provided. Click the “Go” button. It should locate 1 service with a number of methods. Enter “com.adp.hrbws” as the Web reference name and click the “Add Reference” button.

For this tutorial, we will be using the GetEmployeePersonalInfo web service. Once the reference has been added, all of the objects required to send a GetEmployeePersonalInfo request (and all of the other available services) have been created. We are now ready to begin writing code.

Building a simple web service request

Initializing the web proxy client object

To begin using the web service proxy client object that was generated by adding the web reference, we need to import its namespace using project namespaces. This can be done by adding the following line to the Program.cs source file.

```csharp
using TestApp.com.adp.hrbws;
```

Now that we have visibility into this namespace, we have access to the HrbService object which will act as a web service proxy client. To initialize this object, use the following line.

```csharp
HrbService proxy = new HrbService();
```
To simplify the code used in this tutorial, we will not be following good object oriented design in the example code. Go ahead and add the above line to the Main method in the Program class.

In order to take advantage of the WSE 3.0 capabilities, we need to make a small change to the HrbService class definition. Right click on the “HrbService” in the statement above and select “Go To Definition”.

Replace the following line:

```
```

with:

```
public partial class HrbService:Microsoft.WebServices3.WebServicesClientProtocol {
```

This change wraps the proxy using the WSE enhancements, allowing us to add authentication to all of our requests.

**Authentication**

In order to authenticate with HR/Benefits Solution, we will need to add a UsernameToken element to our SOAP header that conforms to the WS-I basic security profile. Luckily, we can take advantage of the functionality in the WSE 3.0 add-on to generate this for us.

In order to build a UsernameToken you must have created the trading partner and operator IDs that were assigned by ADP. For this tutorial, we will use the following credentials:

**Operator:** test  
**Trading Partner:** adp

Import the namespace containing the UsernameToken with the following line:

```
```

The WS-I basic security profile Username token will be formed using a combination of the operator ID and the trading partner ID (e.g. “test@adp”). The Password element is required by the UsernameToken constructor but will not be used for authentication.

```
UsernameToken usernameToken = new UsernameToken("test@adp", "none");
```

Now we need to add this token to the Security header:

```
proxy.RequestSoapContext.Security.Tokens.Add(usernameToken);
```

In order to add the certificates for signing the request and mutual SSL, we will need to import some more packages:

```
```
The next step is to add mutual SSL to the proxy. This is done using the certificate that was exchanged with ADP. There are a variety of ways of loading an X509Certificate, the simplest constructor is below:

```csharp
X509Certificate2 mutualCert = new X509Certificate2("C:\mutual.cert", "password");
```

To add the certificate to the proxy:

```csharp
proxy.ClientCertificates.Add(mutualCert);
```

The final authentication step is to use the signing certificate to sign the request payload:

```csharp
X509Certificate2 signCert = new X509Certificate2("C:\sign.cert", "password");
X509SecurityToken signatureToken = new X509SecurityToken(signCert);
MessageSignature signature = new MessageSignature(signatureToken);
proxy.RequestSoapContext.Security.Elements.Add(signature);
```

**Building the request payload**

Now it is time to populate the request payload. Because we are using a “Get” service, we will be sending a filter in our request payload. If we were using an “Add” or a “Change” web service to modify employee data, we would send an EmployeeKey element and a data element used to modify the existing data (e.g. user account info, corp group).

For this tutorial, we will be using the Companies filter. In .NET, this will be represented as an array of company identifiers (strings).

```csharp
GetEmployeePersonalInfoRequestFilter filter = new GetEmployeePersonalInfoRequestFilter();
filter.Companies = new string[] { "test.com" };
```

- Replace “test.com” with the company identifier for your company in HRB. This will be used in all of the services and may differ from the trading partner ID.
- The Companies filter for “Get” services is the least efficient filter offered. In order to keep response times down for all customers, please request as little information as necessary per request. The Employees filter is more efficient when you only need information for a small number of employees.

Next, we need to create a Request object and apply the filter that we created.

```csharp
GetEmployeePersonalInfoRequest request = new GetEmployeePersonalInfoRequest();
request.Filter = filter;
```

We now have enough information to make our web service call. For the GetEmployeePersonalInfo web service, the result will be an array of EmployeePersonalInfoType objects.

```csharp
EmployeePersonalInfoType[] arPersonalInfo = proxy.GetEmployeePersonalInfo(request);
```
**Parsing the response payload**

Parsing the response is as simple as iterating through the resulting array (in this case an array of EmployeePersonalInfoTypes) and picking out the necessary data.

```csharp
// Loop through each employee in the response payload.
for (int i = 0; i < arPersonalInfo.Length; i++)
{
    String stLastName = arPersonalInfo[i].PersonalInfo.PersonName.LastName;
    String stFirstName = arPersonalInfo[i].PersonalInfo.PersonName.FirstName;
}
```

- For the employee based "Get" services (GetUser, GetEmployeeCorpGroup, etc), each record in the response will include an EmployeeKey element which contains either a National ID, Employee ID, or both. This depends on the company set-up.

- For an "Add" or a "Change web service, each record in the response will include a disposition element which indicates whether the transaction was successful and contains any exception messages that may have been generated during the transaction.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using TestApp.com.adp.hrbws;
namespace TestApp {

class Program {
    static void Main(string[] args) {
        // Create the web service proxy.
        HrbService proxy = new HrbService();

        // Add the Username token.
        UsernameToken usernameToken = new UsernameToken("test@adp","none");
        proxy.RequestSoapContext.Security.Tokens.Add(usernameToken);

        // Add the certificate for mutual SSL.
        X509Certificate2 mutualCert = new X509Certificate2("C:\mutual.cert", "password");
        proxy.ClientCertificates.Add(mutualCert);

        // Sign the message using the signing certificate.
        X509Certificate2 signCert = new X509Certificate2("C:\sign.cert", "password");
        X509SecurityToken signatureToken = new X509SecurityToken(signCert);
        MessageSignature signature = new MessageSignature(signatureToken);
        proxy.RequestSoapContext.Security.Elements.Add(signature);

        // Build a GetEmployeePersonalInfo request using the Companies filter type.
        GetEmployeePersonalInfoRequestFilter filter = new GetEmployeePersonalInfoRequestFilter();
        filter.Companies = new string[] { "test.com" };
        GetEmployeePersonalInfoRequest request = new GetEmployeePersonalInfoRequest();
        request.Filter = filter;

        // Call the web service and store the result in an array of EmployeePersonalInfoType.
        EmployeePersonalInfoType[] arPersonalInfo = proxy.GetEmployeePersonalInfo(request);
// Loop through each employee in the response payload.
for (int i = 0; i < arPersonalInfo.Length; i++)
{
    String stLastName =
        arPersonalInfo[i].PersonalInfo.PersonName.LastName;
    String stFirstName =
        arPersonalInfo[i].PersonalInfo.PersonName.FirstName;
    String stBirthDate =
        arPersonalInfo[i].PersonalInfo.BirthDate.ToShortDateString();
    String stFullName = stLastName + ", " + stFirstName;

    // Write the employees name and birthdate to the console.
    System.Console.Out.WriteLine(stFullName + " - " + stBirthDate);
}

// Display the console until a newline has been entered.
System.Console.In.ReadLine();
**Applying Filters to Get Services**

**Common Filters**

**Employee Key Filter**

The employee key filter is used to for a specific set of one or more employees. An EmployeeKey element contains a unique identifier and a company identifier. The unique identifier consist of a national identifier (e.g. SSN) or an employee identifier.

- The identifier must be both required and unique. Please work with your HR administrator to determine the appropriate identifier that should be sent according to your company set-up.

The example below would return any employees that have an SSN number of 111-11-1111 or an Employee ID of 11111.

```xml
<Filter>
  <Employees>
    <EmployeeKey>
      <Identifier>
        <NationalId idOwner="US">111-11-1111</NationalId>
      </Identifier>
      <CompanyIdentifier>test.com</CompanyIdentifier>
    </EmployeeKey>
    <EmployeeKey>
      <Identifier>
        <EmployeeId>11111</EmployeeId>
      </Identifier>
      <CompanyIdentifier>test.com</CompanyIdentifier>
    </EmployeeKey>
  </Employees>
</Filter>
```

**Company Filter**

The company filter is used to retrieve all employees within a set of one or more companies.

- The company filter is the most resource intensive of all the filters. To avoid timeouts and slower response times, please consider using a more restrictive filter if at all possible. Examples include using the EmployeeKey filter to only request relevant employees or using a combination filter when available.

The example below would return all employees within the test.com company.

```xml
<Filter>
  <Companies>
    <CompanyIdentifier>test.com</CompanyIdentifier>
  </Companies>
</Filter>
```
Last Name Filter

The last name filter is used to retrieve all employees whose last name matches the search criteria. This emulates the functionality of the last name search in the Employee Management Center.

- Two wildcards are supported by this filter, ? and *.

The example below would return all employees whose last name begins with John.

```xml
<Filter>
  <LastNames>
    <LastNameKey>
      <LastName>John*</LastName>
      <CompanyIdentifier>test.com</CompanyIdentifier>
    </LastNameKey>
  </LastNames>
</Filter>
```

Combination Filters

Combination filters are additional filters that can be applied to a Get web service call. A few examples have been included below.

GetEmployeeStatus Example

This example request would return all employees within the test.com company that had a terminated status as of 1/1/2007.

```xml
<GetEmployeeStatusRequest>
  <EffectiveOn>2007-01-01</EffectiveOn>
  <Filter>
    <Companies>
      <CompanyIdentifier>test.com</CompanyIdentifier>
    </Companies>
    <Statuses>
      <StatusCode>Terminated</StatusCode>
    </Statuses>
  </Filter>
</GetEmployeeStatusRequest>
```
GetEmployeeCorpGroup Example
This example request would return the Division and Location corporate groups for all employees within the test.com company as of 3/25/2006.

```xml
<GetEmployeeCorpRequest>
  <EffectiveOn>2006-03-25</EffectiveOn>
  <Filter>
    <Companies>
      <CompanyIdentifier>test.com</CompanyIdentifier>
    </Companies>
    <Types>
      <CorpGroupType>Division</CorpGroupType>
      <CorpGroupType>Location</CorpGroupType>
    </Types>
  </Filter>
</GetEmployeeCorpRequest>
```
Appendix

Related Documents

WS-I Basic Security Profile
http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

XML Schema
http://www.w3.org/2001/XMLSchema

SOAP
http://www.w3.org/TR/soap/

WSDL
http://www.w3.org/TR/wSDL

Example Payloads

GetEmployeeCorpGroup Request

POST /wsi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 2.0.50727.42)
Content-Type: text/xml; charset=utf-8
SOAPAction: "GetEmployeeCorpGroup"
Host: home.eease.com
Content-Length: 1328
Expect: 100-continue
Connection: Keep-Alive
HTTP/1.1 100 Continue

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
            <wsse:Username>test.ecxml@test.com</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">YTSB3aEwvgOcL/QhIiABC2LHjfk=</wsse:Password>
            <wsse:Nonce>df3dL6aaTgzs5T2PujSURA==</wsse:Nonce>
            <wsu:Created>2007-08-02T20:58:07Z</wsu:Created>
        </wsse:UsernameToken>
    </Security>
</soap:Header>
<soap:Body>
  <GetEmployeeCorpGroupRequest>
    <EffectiveOn>2007-08-01</EffectiveOn>
  </GetEmployeeCorpGroupRequest>
</soap:Body>

GetEmployeeCorpGroup Response

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept: application/soap+xml, text/html, image/gif, image/jpeg, */*; q=.2, */*;
q=.2
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 584
Date: Thu, 02 Aug 2007 20:52:47 GMT

<env:Envelope xmlns:env="http://www.w3.org/2003/05soap-envelope">
  <env:Body>

  </env:Body>
</env:Envelope>
HireEmployee Request

POST /wsi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 2.0.50727.832)
Content-Type: text/xml; charset=utf-8
SOAPAction: "HireEmployee"
Host: home.eease.com
Expect: 100-continue
Connection: Keep-Alive

HTTP/1.1 100 Continue

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
       wsu:Id="SecurityToken-2546a2b8-64ce-4bad-9493-f7c4b3478630"
       xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <wsse:Username>test.ecxml@test.com</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-username-token-profile-1.0#PasswordDigest">J8a3JsZjr34sXyZd5SkWneNrwY194=</wsse:Password>
        <wsse:Nonce>res8w6vaYmWyyE32augdd5+A==</wsse:Nonce>
        <wsu:Created>2007-08-2T20:11:16Z</wsu:Created>
      </wsse:UsernameToken>
    </Security>
  </soap:Header>
  <soap:Body>
    <HireEmployeeRequest>
      <Employee>
        <EmployeeKey>
          <Identifier>
            <NationalId idOwner="US">111-11-1111</NationalId>
          </Identifier>
          <CompanyIdentifier>test.com</CompanyIdentifier>
        </EmployeeKey>
        <HireDate>2007-08-01</HireDate>
        <Reason>Import created action</Reason>
        <StatusClassification>Regular</StatusClassification>
        <FullPartTime>Full-time</FullPartTime>
        <PersonalInfo>
          <PersonName>
            <FirstName>Joe</FirstName>
            <LastName>Johnson</LastName>
          </PersonName>
          <Gender>Male</Gender>
        </PersonalInfo>
      </Employee>
    </HireEmployeeRequest>
  </soap:Body>
</soap:Envelope>
<BirthDate>1950-01-01</BirthDate>
</PersonalInfo>
<WorkInfo>
<EmployeeID>11111111</EmployeeID>
<EmailAddress>jjohnson@test.com</EmailAddress>
</WorkInfo>
<CorpGroups>
<Division>
<CorpGroupType>Division</CorpGroupType>
<CorpGroupName>Corporate</CorpGroupName>
<CompanyIdentifier>test.com</CompanyIdentifier>
</Division>
<Location>
<CorpGroupType>Location</CorpGroupType>
<CorpGroupName>Baltimore</CorpGroupName>
<CompanyIdentifier>test.com</CompanyIdentifier>
</Location>
<Class>
<CorpGroupType>Class</CorpGroupType>
<CorpGroupName>Benefits Eligible</CorpGroupName>
<CompanyIdentifier>test.com</CompanyIdentifier>
</Class>
<Department>
<CorpGroupType>Department</CorpGroupType>
<CorpGroupName>Sales</CorpGroupName>
<CompanyIdentifier>test.com</CompanyIdentifier>
</Department>
<Union>
<CorpGroupType>Union</CorpGroupType>
<CorpGroupName>N/A</CorpGroupName>
<CompanyIdentifier>test.com</CompanyIdentifier>
</Union>
</CorpGroups>
<PayGroup>
<PayGroupName>Weekly</PayGroupName>
<CompanyIdentifier>test.com</CompanyIdentifier>
</PayGroup>
</Employee>
</Employees>
</HireEmployeeRequest>
</soap:Body>
</soap:Envelope>
HireEmployee Response

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept: application/soap+xml, text/html, image/gif, image/jpeg, */*; q=.2, */*
q=.2
Content-Type: application/soap+xml;charset=utf-8
Content-Length: 401
Date: Tue, 21 Aug 2007 20:06:01 GMT

<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <HireEmployeeResponse>
      <EmployeeStatus>
        <Identifier>
          <NationalId idOwner="US">111-11-1111</NationalId>
        </Identifier>
        <CompanyIdentifier>test.com</CompanyIdentifier>
      </EmployeeKey>
      <EmployeeDisposition>
        <Successful>true</Successful>
      </EmployeeDisposition>
    </EmployeeStatus>
  </HireEmployeeResponse>
</env:Body>
</env:Envelope>