Configuration Web Services for .NET Framework

Programmer’s Guide

October 2014

This guide describes how to create a client for the Configuration Web Services with the .NET framework and C# language.
About Five9

Five9 is the leading provider of cloud contact center software, bringing the power of the cloud to thousands of customers and facilitating more than three billion customer interactions annually. Since 2001, Five9 has led the cloud revolution in contact centers, delivering software to help organizations of every size transition from premise-based software to the cloud. With its extensive expertise, technology, and ecosystem of partners, Five9 delivers secure, reliable, scalable cloud contact center software to help businesses create exceptional customer experiences, increase agent productivity and deliver tangible results. For more information visit www.five9.com.

Trademarks

Five9®
Five9 Logo
Five9® SoCoCare™
Five9® Connect™
Contents

What’s New ..................................................................................................................... iv

Chapter 1
Configuration Web Services for .NET Framework ........................................ 1
Audience ..................................................................................................................... 1
Web Services Platform ............................................................................................ 1
Creating a .NET Application ................................................................................. 2

Chapter 2
Creating Code ......................................................................................................... 4
Enabling HTTP Basic Authentication ................................................................... 4
  C# Example: WsAdminClient.cs ......................................................................... 4
  Visual Basic Example: client.vb ......................................................................... 5
Using Generated Code ......................................................................................... 6

Appendix A
Source Code .......................................................................................................... 8
Program.cs ............................................................................................................ 8
asyncAddRecordsToList.cs .................................................................................... 9
Program.vb ........................................................................................................... 13
CreateUser.vb ...................................................................................................... 14
What’s New

This table lists the changes made in the last releases of this document:

<table>
<thead>
<tr>
<th>Release</th>
<th>Changes</th>
</tr>
</thead>
</table>
| October 2014 | • Added a note about the examples in the guide.  
• Updated Creating a .NET Application.  
• Added a note about finding additional code examples at GitHub. |
| September 2013 | • Re-branded the guide with the new Five9 logo.  
• Added the asyncAddRecordsToList.cs example. |
| June 2012 | Version 2 of the Web Services API WSDL is now available:  
The original version of the file remains at this location:  
Chapter 1
Configuration Web Services for .NET Framework

This guide describes how to create a client for the Configuration Web Services with the .NET framework version 4.5 and newer (Visual Studio 2012 and 2013).

Important The code snippets in this guide are examples only. You cannot compile the snippets because they are incomplete.

Audience

This guide is intended for developers who want to create contact-center applications. Developers must understand these technologies:

- Client-server architecture and Web Services
- SOAP, HTTP, and XML
- .NET
- C# and Visual Basic
- Overall call-center integration and configuration

Web Services Platform

The Web Services API contains the XML-encoded SOAP methods required to communicate with your client application. The WSDL file is located at this URL:


version: your version of the WSDL, for example: v2.
Five9username: user name of the administrator.

To ensure that connections are secure, send all requests by Transport Layer Security protocol (HTTPS) or VPN (IPSec or SSH) to this URL:

https://api.five9.com/wsadmin[/<version>]/AdminWebService
Creating a .NET Application

You need to download the WSDL and add a service reference to a project.

1. In a browser, locate the WSDL for your version of the API:
   https://api.five9.com/wsadmin[/<version>/]
   AdminWebService?wsdl&user=<Five9username>

2. Save it to your computer.

   **Important** Because you manually downloaded the WSDL, further updates to
   the Five9 WSDL will not automatically update your implementation. Instead,
   you will need to manually download the WSDL again to add the service
   reference.

3. In Visual Studio, create a console application.

4. Select **Project > Add Service Reference**.

5. Add this information:
   - **Address**: absolute path to the WSDL that you downloaded
   - **Namespace**: your name space
6 Click Go.
The WSDL appears in the Services area.

7 Click OK.
The service reference appears in the Solution Explorer.
Chapter 2

Creating Code

This chapter describes the code that you need to prepare.

To download additional .NET examples, see the Five9 Developer Program at GitHub.

Enabling HTTP Basic Authentication

Before using the classes generated by the WSDL, you need to override the method that handles Basic Authentication. This method, called GetWebRequest, is located in the WsAdminService class. To do so, add one of the following classes to your project path, ensuring that you change the name space to that of your generated classes.

Another solution is to derive from WsAdminService a class that contains an overriding method as shown in the examples below. This class should be in the same name space as WsAdminService.

C# Example: WsAdminClient.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;

// Change the name space to the location of WsAdminService.
namespace WsApi.WsApi
{
    public partial class WsAdminService
    {
        protected override System.Net.WebRequest GetWebRequest(Uri uri)
        {
            HttpWebRequest request;
            request = (HttpWebRequest)base.GetWebRequest(uri);
```
if (PreAuthenticate)
{

    NetworkCredential networkCredentials =
        Credentials.GetCredential(uri, "Basic");

    if (networkCredentials != null)
    {
        byte[] credentialBuffer = new UTF8Encoding().GetBytes(
            networkCredentials.UserName + ":" +
            networkCredentials.Password);
        request.Headers["Authorization"] =
            "Basic " + Convert.ToBase64String(credentialBuffer);
    }
    else
    {
        throw new ApplicationException("No network credentials");
    }
}
}

return request;

Visual Basic Example: client.vb

Imports System.Net
Imports System.Text

' Change the name space to the location of WsAdminService.
Namespace api.five9.com

' This class is derived from WsAdminService to avoid overriding
' GetWebRequest each time after WEB Reference update.
'
' It is a good place to add a utility method service similar to
' SetupCredentials below.
'
Partial Public Class WsAdminClient
    Inherits WsAdminService

    ' Need to avoid problem with HTTP Basic Authentication
    Protected Overrides Function GetWebRequest(ByVal uri As Uri) As System.Net.WebRequest

        Dim request As HttpWebRequest
        request = DirectCast(MyBase.GetWebRequest(uri), HttpWebRequest)
If PreAuthenticate Then

    Dim networkCredentials As NetworkCredential = Credentials.GetCredential(uri, "Basic")

    If networkCredentials IsNot Nothing Then

        Dim credentialBuffer As Byte() = New UTF8Encoding().GetBytes(networkCredentials.UserName + ":" + networkCredentials.Password)

        request.Headers("Authorization") = "Basic " + Convert.ToBase64String(credentialBuffer)

    Else

        Throw New ApplicationException("No network credentials")

    End If

End If
End If
Return request
End Function

' Sub to set up credentials to access service.
Public Sub SetupCredentials(ByVal user As String, ByVal password As String)

    Dim serviceCredentials As CredentialCache = New CredentialCache()
    Dim netCredentials As NetworkCredential = New NetworkCredential(user,password)

    serviceCredentials.Add(New Uri(Url), "Basic", netCredentials)
    Credentials = netCredentials.GetCredential(New Uri(Url), "Basic")

    PreAuthenticate = True

End Sub
End Class
End Namespace

Using Generated Code

The code contains these classes:

- **Main wrapper class:** WsAdminService
- **Classes, such as getSkills, that hold operation parameters**
• Service-defined classes, such as `userInfo`

This example describes how to specify the pattern of a list name to obtain a list of dialing lists:

1. Create an instance of the wrapper class.
   ```csharp
   // Create instance of Web service proxy.
   WsAdminService service = new WsAdminService();
   ```

2. Set up user credentials.
   ```csharp
   // Set up credentials.
   System.Net.CredentialCache serviceCredentials =
   new System.Net.CredentialCache();
   NetworkCredential netCredentials =
   new NetworkCredential("username", "password");
   serviceCredentials.Add(new Uri(service.Url), "Basic",
   netCredentials);
   service.Credentials =
   netCredentials.GetCredential(new Uri(service.Url), "Basic");
   service.PreAuthenticate = true;
   ```

3. Create the parameter object for the operation.
   ```csharp
   // Create and initialize getListInfo parameter object. getListsInfo
   paramObj = new getListsInfo(); paramObj.listNamePattern = ".*"; // Select all lists.
   ```

4. Call the operation.
   ```csharp
   // Call getListInfo.
   listInfo[] result = service.getListsInfo(paramObj);
   ```

For the complete source code, see `Program.cs`. Not all operations return results as plain as `getListsInfo`. For example, `addRecordToListResponse` contains `listImportResult` in the `addRecordToListResponsInstance.@return` property:

```csharp
addRecordToListResponse response =
service.addRecordToList(addRecordToListParam);
listImportResult importResult = response.@return;
```
or:

```csharp
listImportResult importResult =
service.addRecordToList(addRecordToListParam).@return;
```

5. To set the enumerated parameters in the classes that contain them, such as `listUpdateSettings`, specify the parameter value and set it to true:
   ```csharp
   settings.crmAddMode = crmAddMode.ADD_NEW;
   settings.crmAddModeSpecified = true; // set true. Otherwise, settings.crmAddMode is null.
   ```
Appendix A

Source Code

This appendix contains examples of source code that you can use in your implementation.

Program.cs
asyncAddRecordsToList.cs
Program.vb
CreateUser.vb

Program.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using WebApiSampleClient.com.five9.api; // For WsAdminService
using System.Net; // For NetworkCredential

namespace WebApiSampleClient
{
    class Program
    {

        static listInfo[] getListInfosByPattern(WsAdminService service, String pattern)
        {
            // Create and initialize getListInfo parameter object.
            getListInfo paramObj = new getListInfo();
            paramObj.listNamePattern = pattern; // select all lists

            // Call getListInfo.
            listInfo[] result = service.getListsInfo(paramObj);

            return result;
        }

        static void Main(string[] args)
        {
            // Create instance of Web service proxy.
        }
    }
}
WsAdminService service = new WsAdminService();

    // Set up credentials.
    System.NetCredentialCache serviceCredentials =
        new System.NetCredentialCache();
    NetworkCredential netCredentials =
        new NetworkCredential("username", "password");
    serviceCredentials.Add(new Uri(service.Url), "Basic", netCredentials);

    service.Credentials =
        netCredentials.GetCredential(new Uri(service.Url), "Basic");
    service.PreAuthenticate = true;

    // Call operation.
    listInfo[] lists = getListInfosByPattern(service, ".*");

    // Debug output.
    foreach (listInfo list in lists)
    {
        System.Console.WriteLine("List \\
"%s\\n"	 has size: \<%d\>",
            list.name, list.size);
    }

    // Stop after execution.
    System.Console.Write("\nPress Enter to close");
    System.Console.ReadLine();
}
static WsAdminService service;

public static void asyncAddRecordsToList()
{
    fieldEntry number1 = new fieldEntry();
    //the first column should have columnNumber of 1, not 0:
    number1.columnNumber = 1;
    number1.fieldName = "number1";
    number1.key = true;

    fieldEntry last_name = new fieldEntry();
    last_name.columnNumber = 2;
    last_name.fieldName = "last_name";
    last_name.key = false;

    fieldEntry first_name = new fieldEntry();
    first_name.columnNumber = 3;
    first_name.fieldName = "first_name";
    first_name.key = true;

    listUpdateSettings updateParams = new listUpdateSettings()
    {
        fieldsMapping = new fieldEntry[] { number1, last_name,
        first_name },
        //skipHeaderLine = false,
        //Not applicable to method. Only addToListCsv
        //separator = ",",
        //Not applicable to method. Only in addToListCsv.
        //reportEmail = "email@dsf.com",
        //Not applicable to method. Only in addToList.
        callNowMode = callNowMode.NONE,
        //If set to ANY, all records are added to ASAP queue.
        //callNowColumnNumber = 5,
        //To add only some records to ASAP queue, a separate
        column in importData can act as a flag.
        //callNowColumnNumberSpecified = true,
        //set if callNowColumnNumber is specified.
        //callTime = 333333,
        //Time to call all the records in UNIX time (number of
        millisecords since Jan 1, 1970).
        //callTimeSpecified = true,
        //set if callTime is specified.
        //callTimeColumnNumber = 6,
        //to call records at different times, use a separate
        column in importData to specify the time.
        //callTimeColumnNumberSpecified = true,
        //set if callTimeColumnNumber is specified.
        callNowModeSpecified = true, // set true. Otherwise is null.
        cleanListBeforeUpdate = false,
        crmAddMode = crmAddMode.ADD_NEW,
asyncAddRecordsToList param = new asyncAddRecordsToList()
{
    listName = "asdf",
    listUpdateSettings = updateParams,
    importData = new string [][]{
        {"5551234458", "Billy", "Butch"},
    }
};

asyncAddRecordsToListResponse result;
try
{
    result = service.asyncAddRecordsToList(param);
}
catch (Exception)
{
    throw;
}
importIdentifier ID = result.@return;
bool importRunning = true;
while (importRunning == true)
{
    isImportRunningResponse importReturn = service.isImportRunning(new
    isImportRunning()
    {
        identifier = ID,
        waitTime = 5,
        waitTimeSpecified = true,
    });
    importRunning = importReturn.@return;
}
try
{
    listImportResult importResult = service.getListImportResult(new
    getListImportResult
    {
        identifier = ID,
    }).@return;
    Console.WriteLine("Records added to ASAP queue : " +
importResult.callNowQueued);
Console.WriteLine("Records added to CRM : " +
    importResult.crmRecordsInserted);
Console.WriteLine("Records in CRM updated : " +
    importResult.crmRecordsUpdated);
if (!string.IsNullOrEmpty(importResult.failureMessage))
    Console.WriteLine("Error message : " +
    importResult.failureMessage);
Console.WriteLine("Records deleted from list : " +
    importResult.listRecordsDeleted);
Console.WriteLine("Records inserted in list : " +
    importResult.listRecordsInserted);
Console.WriteLine("Records in list already have the same key : " +
    importResult.uploadDuplicatesCount);
Console.WriteLine("Records not inserted due to error : " +
    importResult.uploadErrorsCount);
if (importResult.warningsCount.Count() > 0)
    Console.WriteLine("Warnings:");
    foreach (basicImportResultEntry warning in
        importResult.warningsCount)
    {
        Console.WriteLine(warning.key + " : " + warning.value);
    }
    catch (Exception)
    {
        throw;
    }
getListImportResult getResultP = new getListImportResult()
    {
        identifier = ID,
    };

//
static void Main(string[] args)
{
  // Create instance of Web service proxy.
  service = new WsAdminService();

  // Setup credentials
  System.Net.CredentialCache serviceCredentials = new
      System.Net.CredentialCache();
  NetworkCredential netCredentials = new NetworkCredential("your_
      login", "your_
      password");
serviceCredentials.Add(new Uri(service.Url), "Basic", netCredentials);

service.Credentials = netCredentials.GetCredential(new Uri(service.Url), "Basic");

service.PreAuthenticate = true;

asyncAddRecordsToList();

System.Console.Write("Press any key to exit...");
System.Console.Read();

// Stop after execution
System.Console.Write("\n");
}
}

Module Program

Sub GetUsers(ByRef service As WsAdminService, ByVal pattern As String)
' Create argument instance to be passed to the service operation
Dim getUsersInfoArg As getUsersInfo = New getUsersInfo()

' Define arguments withing created instance
getUsersInfoArg.userNamePattern = pattern

' Call operation and get result
Dim result() As userInfo = service.getUsersInfo(getUsersInfoArg)

' Process result (out put it)
If result.Length > 0 Then
Dim i As Integer = 0
For Each user As userInfo In result
    Console.WriteLine(i.ToString + ". " + user.generalInfo.userName)
    i = i + 1
Next
Else
    Console.WriteLine("The result for pattern " + pattern + " is empty. ")
End If

Imports WebApi.api.five9.com
Imports System.Net
Imports System.Net.Security
End If

End Sub

Sub Main()
    ' Create proxy instance for Web service. For WsAdminClient source, see Client.vb.
    Dim service As WsAdminClient = New WsAdminClient()

    ' Set user name and password.
    ' See code of SetupCredentials in WsAdminClient class.
    service.SetupCredentials("test", "test")

    ' Fetch all users for domain.
    GetUsers(service, ".*")

    ' Wait until user presses Enter.
    Console.Write("Press Enter to exit")
    Console.ReadLine()

End Sub

End Module

CreateUser.vb

Imports ConsoleApplication1.com
Imports System.Net
Imports System.Net.Security
Imports ConsoleApplication1.com.five9.api

Module Program

    Sub CreateUser(ByRef service As five9.api.WsAdminService)
        ' Create argument instance to be passed to service operation.
        ' Define General User Information, such as Name, Username, Password.
        Dim NewUserGeneral As New userGeneralInfo
        NewUserGeneral.fullName = "John Smith"
        NewUserGeneral.userName = "Smithy"
        NewUserGeneral.EMail = "something@somedomain.com"
        NewUserGeneral.canChangePassword = True
        NewUserGeneral.canRunReports = False
        NewUserGeneral.mustChangePassword = True
        NewUserGeneral.password = "some_password"

        Dim NewRoles As New userRoles
' To give Agent role, add 1 or more permissions:

Dim NewPermCallForwarding As New agentPermission
NewPermCallForwarding.type = agentPermissionType.CallForwarding
NewPermCallForwarding.typeSpecified = True
NewPermCallForwarding.value = True

Dim NewPermCreateConference As New agentPermission
NewPermCreateConference.type = agentPermissionType.CreateConference
NewPermCreateConference.typeSpecified = True
NewPermCreateConference.value = True

Dim myAgent As New agentRole
myAgent.permissions = {NewPermCallForwarding, NewPermCreateConference}
myAgent.alwaysRecorded = False
myAgent.attachVmToEmail = False

NewRoles.agent = myAgent

' To give Admin role, add 1 or more permissions:

Dim admper1 As New adminPermission
admper1.type = adminPermissionType.FullPermissions
admper1.typeSpecified = True
admper1.value = True

NewRoles.admin = {admper1}

' To give Supervisor role, add 1 or more permissions:

Dim superP As New supervisorPermission
superP.type = supervisorPermissionType.Agents
superP.typeSpecified = True
superP.value = True

NewRoles.supervisor = {superP}

' specify 1 or more user-skill relations:

Dim NewSkill As New userSkill
NewSkill.skillName = "some_skill"
NewSkill.userName = "Smithy"
NewSkill.level = 1
' Parameters for createUser are generalInfo, roles, and skills:

Dim NewUser As New userInfo
NewUser.generalInfo = NewUserGeneral
NewUser.roles = NewRoles
NewUser.skills = {NewSkill}

Dim NewUserCreate As New createUser
NewUserCreate.userInfo = NewUser

' Execute and check for errors:

Try
    Dim result As createUserResponse = service.createUser(NewUserCreate)
    Console.WriteLine(result.return.generalInfo.fullName)
    Console.WriteLine(myException.Message)
End Try

Sub Main()
    ' Create proxy instance for Web service. See WsAdminClient source.
    Dim service As WsAdminClient = New WsAdminClient()

    ' Set user name and password
    ' See code of SetupCredentials in WsAdminClient class
    service.SetupCredentials("your_logon", "your_password")

    ' Call this method:
    CreateUser(service)

    ' Wait until user presses Enter.
    Console.WriteLine("Press Enter to exit.")
    Console.ReadLine()
End Sub
End Module