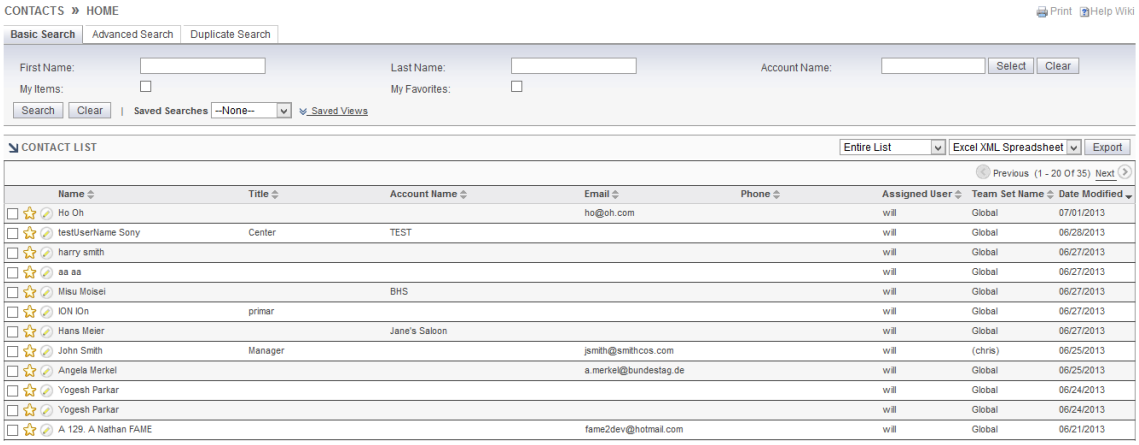
Using iLove SharePoint Web Services Workflow Action

This guide describes the steps to create a workflow that will add some information to “Contacts” in CRM.
As an example, we will use demonstration site http://demo.splendidcrm.com with login “will” and password “will”.



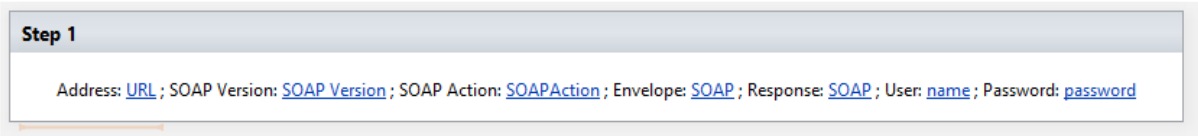
We will add new contact with person’s name, last name and email.

NOTE:
iLove SharePoint workflow action for calling web services is working only with “http” by default.
In order to make it working with “https”, you will need to change its code slightly (this step is not necessary – see appendix at the end of this guide).

iLove SharePoint workflow actions can be downloaded from the following link:
http://ilovesharepoint.codeplex.com/

Creating a new workflow

Step 1. Add **Call a Web Service** from the **Action** menu:



This will be our workflow with only one action. It has the following fields:

URL	Target site URL
SOAP Version	Version of SOAP request
SOAP Action	Action of used SOAP request
SOAP	XML SOAP envelope
name	Account name
password	Account password

Step 2. Fill the action with data:

NOTE: Above shown action in single.

1) http://demo.splendidcrm.com/soap.asmx - it is site URL (http://demo.splendidcrm.com) + SOAP request receiver (soap.asmx);

NOTE:
List of SOAP requests is also located at the following URL: http://demo.splendidcrm.com/soap.asmx

NOTE:
Every CRM has a list of SOAP requests in request receiver, but requests structure may slightly differ.

2) SOAP Version – "SOAP 1.1";

3) SOAP Action – add the following action:
http://www.sugarcrm.com/sugarcrm/create_contact - it is site URL + SOAP action (sugarcrm/create_contact);

4) SOAP – add the following envelope:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.sugarcrm.com/sugarcrm"
xmlns:types="http://www.sugarcrm.com/sugarcrm/encodedTypes"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <tns:create_contact>
      <user_name xsi:type="xsd:string">will</user_name>
      <password xsi:type="xsd:string">18218139eec55d83cf82679934e5cd75</password>
      <first_name xsi:type="xsd:string">Luke</first_name>
      <last_name xsi:type="xsd:string">Walker</last_name>
      <email_address xsi:type="xsd:string">luke@wal.com</email_address>
    </tns:create_contact>
  </soap:Body>
</soap:Envelope>
```

This XML envelope will create a new contact with the following entries:

```
<first_name xsi:type="xsd:string">Luke</first_name>
<last_name xsi:type="xsd:string">Walker</last_name>
<email_address xsi:type="xsd:string">luke@wal.com</email_address>
```

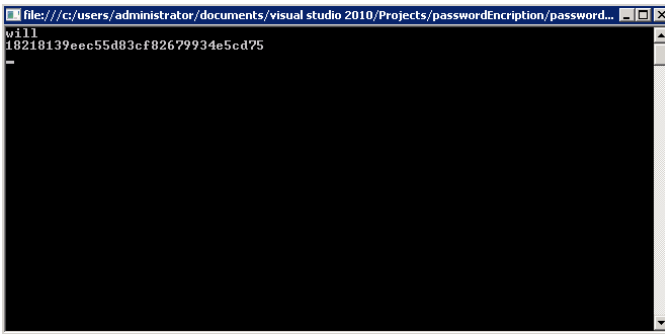
Name	Luke
Last name	Walker
E-mail address	luke@wal.com

There is no need to create a separate login request before this one. It will authenticate user before creating a new contact:

```
<user_name xsi:type="xsd:string">will</user_name>

<password xsi:type="xsd:string">18218139eec55d83cf82679934e5cd75</password>
```

NOTE:
The account we use in this example has password “will”, but in envelope it has been used combination of symbols.
This is because password must be encrypted with MD5.
The action must be done manually: in this case will be used console application that will scan user entered password and return its encrypted version.
Here is the result on encrypting “**will**”:



Full code of this console application:

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

namespace passwordEncription
{
    class Program
    {
        static void Main(string[] args)
        {
            string pass = Console.ReadLine();
            string rez = GetMD5Hash(pass, false);
            Console.WriteLine(rez);
            Console.ReadLine();
        }

        public static string GetMD5Hash(string value, bool upperCase)
        {
            // Instantiate new MD5 Service Provider to perform the hash
            System.Security.Cryptography.MD5CryptoServiceProvider md5ServiceProdivder
 = new System.Security.Cryptography.MD5CryptoServiceProvider();

            // Get a byte array representing the value to be hashed and hash it
            byte[] data = System.Text.Encoding.ASCII.GetBytes(value);
            data = md5ServiceProdivder.ComputeHash(data);

            // Get the hashed string value
            StringBuilder hashedValue = new StringBuilder();
            for (int i = 0; i < data.Length; i++)
                hashedValue.Append(data[i].ToString("x2"));

            // Return the string in all caps if desired
            if (upperCase)
                return hashedValue.ToString().ToUpper();

            return hashedValue.ToString();
        }
    }
}
```

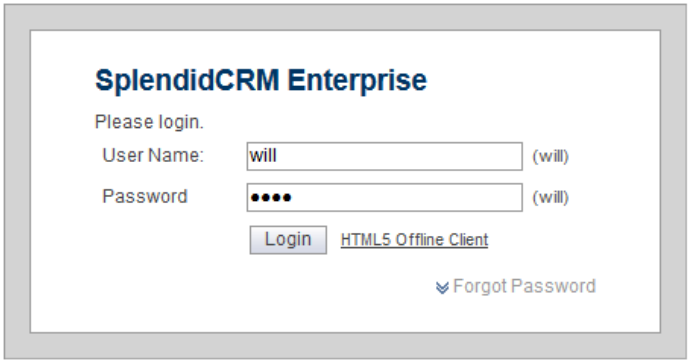
5) Response, name and password fields stay blank because we do not need to save returned response. Name and password have been sent in request's envelope.

Step 3. Save workflow and deploy it.

Checking results

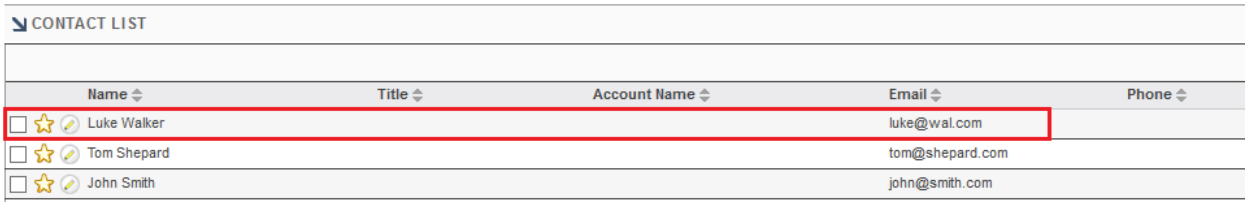
Step 1. Run the workflow manually or from any kind of event.

Step 2. Log in to http://demo.splendidcrm.com using account name “will” and password “will”:



Step 3. Go to the **Contacts** section:

Step 4. Check results:



As can be seen, the new contact has been added.

Appendix

Changing workflow action code to make it work with HTTPS

The problem is in SSL Certificate checking while connecting to the server using this workflow action.
This problem can be solved by adding Certificate Validation to code in "**CallWebServiceActivity.cs**" class:

```
ServicePointManager.ServerCertificateValidationCallback = new RemoteCertificateValida
tionCallback(delegate { return true; });
```

This code must be placed before any attempting to connect the server. Workflow "**Execute**" function:

```
protected override ActivityExecutionStatus Execute(ActivityExecutionContext execution
Context)
        {
            SPSecurity.RunWithElevatedPrivileges(delegate()
            {
                WebClient webClient = new WebClient();

                if(SoapVersion==soap12)
                    webClient.Headers[HttpRequestHeader.ContentType] = "application/s
oap+xml; charset=utf-8";
                else
                    webClient.Headers[HttpRequestHeader.ContentType] = "text/xml; cha
rset=utf-8";

                webClient.Headers["SOAPAction"] = Action;
                if (String.IsNullOrEmpty(User))
                {
                    webClient.UseDefaultCredentials = true;
                }
                else
                {
                    webClient.Credentials = new NetworkCredential(User, Password);
                }

                Activity parent = executionContext.Activity;
                while (parent.Parent != null)
                {
                    parent = parent.Parent;
                }

                Response = webClient.UploadString(Address, Helper.ProcessStringField(
Envelope, parent, this.__Context));
            });

            return ActivityExecutionStatus.Closed;
        }
```

Certificate Validation must be placed at the start of function, in this case after "**webClient**" declaration:

```
SPSecurity.RunWithElevatedPrivileges(delegate()
            {
                WebClient webClient = new WebClient();
                ServicePointManager.ServerCertificateValidationCallback = new RemoteC
ertificateValidationCallback(delegate { return true; });

                if(SoapVersion==soap12)
                    webClient.Headers[HttpRequestHeader.ContentType] = "application/s
oap+xml; charset=utf-8";
                else
                    webClient.Headers[HttpRequestHeader.ContentType] = "text/xml; cha
rset=utf-8";
```

Add reference for "**RemoteCertificateValidationCallback**":

```
using System.Net.Security;
```