

**Write data from your external  
systems using the Salesforce API**

**Oct 2011**



**A Bluewave Group Company**



The Software Solution Experts

<b>Writing Data to Salesforce from Domino using the Salesforce Web Services API .....</b>	<b>3</b>
The steps to complete the above task are outlined below.. .....	3
Obtain the WSDL file for Force.com .....	3
Import the WSDL file into your platform .....	4
Create the Agent .....	6



The Software Solution Experts

## Writing Data to Salesforce from Domino using the Salesforce Web Services API

Clearly Salesforce has become a very popular platform for organisations to track CRM related content and beyond with the Force.com PaaS. However there was life before Salesforce and organisations always require the capacity to exchange data between older and newer systems.

This article is written with this need in mind and specifically uses Notes Domino as the source database and a Salesforce Object as the target. The example in this article takes data entered into a notes form that is browser based and writes this data to a Salesforce Contact object.

The normal health warnings apply in the Bluewave Technology Ltd take no responsibility for its use and offer no warrantee with regard to the content. All usage is done at the users own risk.

The scenario here is that a user registers for a seminar using a Domino Form on the web, the data is subsequently submitted to Domino and written directly to a Salesforce Contact object at the same time. I have deliberately kept this example simple, in that we don't create an accompanying Account object to associate with the Contact, however this would be simple enough to add. Also the web form that takes in the data is not dealt with, suffice to say it is a standard Domino web form that submits data to the backend using a standard submit.

**The steps to complete the above task are outlined below..**

- Obtain the WSDL file for Force.com
- Import the WSDL file into your platform
- Write the agent
  - Take the data from the Web Form
  - Write the data to Salesforce Platform

### Obtain the WSDL file for Force.com

Details on how to do this are contained in the apex-api.pdf which can be obtained at <http://wiki.developerforce.com/index.php/Documentation#API>. But a quick description is outlined below..



## The Software Solution Experts

API

### Force.com Web Services API Developer's Guide

Version 23.0 for Winter '12: [HTML](#) | [PDF](#) | [ZIP](#) | [What's New](#) | [Samples](#) | [Earlier Versions](#)

(Audience: Developers) Reference for the SOAP-based Force.com API.

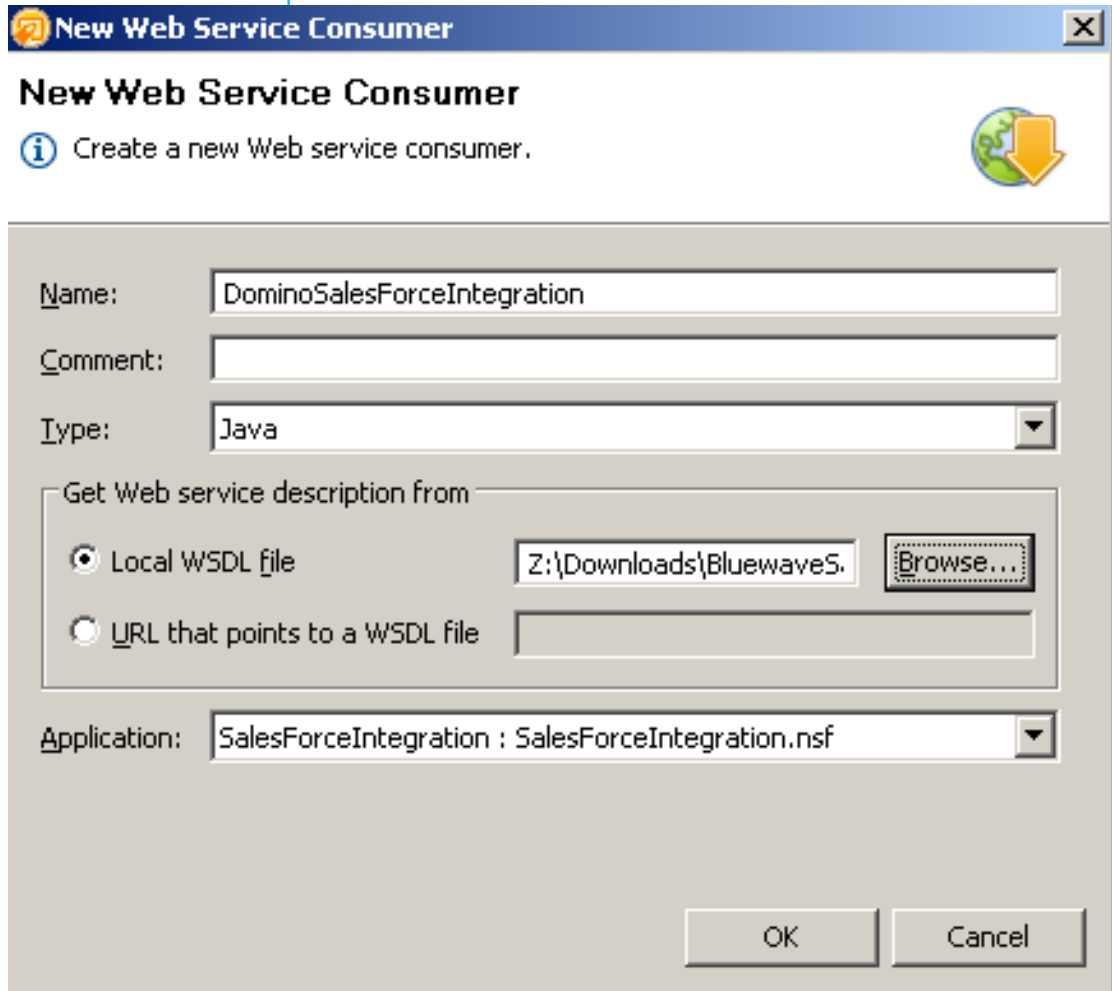
To access the Force.com Web service, you need a Web Service Description Language (WSDL) file. The WSDL file defines the Web service that is available to you. Your development platform uses this WSDL to generate an API to access the Force.com Web service it defines. You can either obtain the WSDL file from your organization's Salesforce administrator or you can generate it yourself if you have access to the WSDL download page in the Salesforce user interface. You can navigate to the most recent WSDL for your organization by clicking *Your Name* > Setup > Develop > API.

### Import the WSDL file into your platform

Once you have the WSDL file, you need to import it into your development platform so that your development environment can generate the necessary objects for use in building client Web service applications.

We are using Domino and we have the facility to create the appropriate Objects in Java that we can then use to write the data to Salesforce. This example uses Java, I have not tested with LotusScript.

In the Domino Designer client create a web service consumer. The following dialogue will appear



**Fig 1. New Domino Web Service Consumer Dialogue**

**Name :** The name you refer to the Web Service Consumer within the Domino application.

**Comment :** Any comments you may have

**Type:** Language to be used (Our example uses Java)

**Get WSDL from :** Point to the location of the WSDL

**Application :** Application the Web Service Consumer applies To.

Having selected OK Domino will proceed to create the Objects that are a reflection of the Salesforce API, which we can use in our agent to write to the Salesforce system.

Once Domino has processed the WSDL file, the following screen will appear.

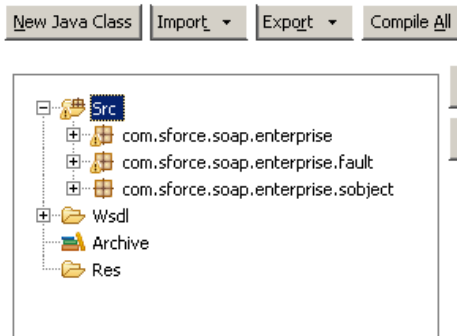


## The Software Solution Experts

**BWEnt**

in Sametime/BlueWave/Bluewave/Web.nsf

### Web Service Consumer Contents



### Editing the Web Service Consumer

This Java Web Service Consumer tab contains the contents of the Java web service consumer. Here, you can create or import files, and select files to edit.

To edit a file, double-click it in the list; the file opens for editing in a separate tab.

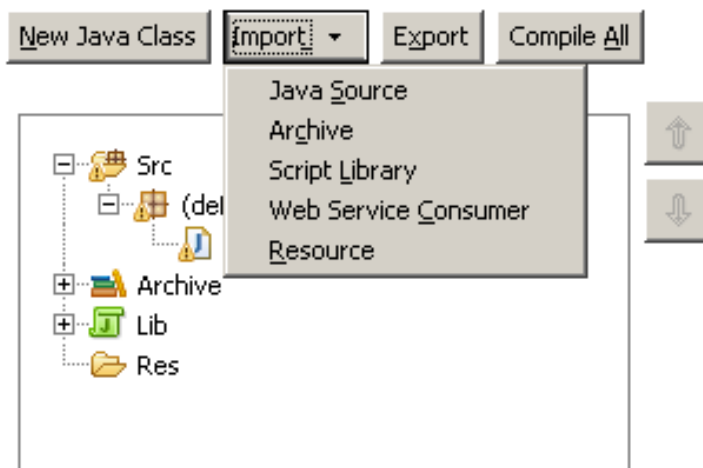
Keep this Java Web Service Consumer tab open as you edit the files, save and close it when you finish editing.

**Fig 2. Post Web Service Consumer Screen**

## Create the Agent

Remember that the Agent is written in Java, so you can make all required external JAR file, Web Consumers etc, using the Import option in the Agent's properties. Be sure to choose the Web Consumer you just created as well as any other libraries you may need.

### Agent Contents



### Editing the Agent

This Java Agent tab contains the contents of the agent. Here, you can create or import files to edit.

To edit a file, double-click it in the list; the file opens for editing in a separate tab.

Keep this Java Agent tab open as you edit the files, save and close it when you finish editing.

**Fig 3. Import any external resources required for Agent**

You are now in a position to write the agent.



### The Software Solution Experts

The import statements at the start, give access to the Domino Library and the Salesforce api. The latter being available as we imported the previously created web services consumer in the previous step. The former available by default.

```
import lotus.domino.*;
import com.sforce.soap.partner.*;
import com.sforce.ws.*;
import com.sforce.soap.partner.subject.SObject;
```

The next few lines establish a connection to Salesforce using your Salesforce name and password(plus security Token).. The last line creates a list of list of Standard Objects based on the number of documents in the Domino View.

```
public class JavaAgent extends AgentBase {
    public void NotesMain() {
        try {
            Session session = getSession();
            AgentContext agentContext = session.getAgentContext();
            ConnectorConfig config = new ConnectorConfig();
            config.setUsername("Yoursalesforcename");
            config.setPassword("yoursalesforcepasswordandsecuritytoken");
            Database db = agentContext.getCurrentDatabase();
            View view = db.getView("Registrations");
            int noofdocs = view.getEntryCount();
            lotus.domino.Document tmpdoc;
            lotus.domino.Document doc = view.getFirstDocument();
            PartnerConnection connection = Connector.newConnection(config);
            // SObject contact = new SObject();
            SObject[] records = new SObject[noofdocs];
```

The next While loop goes through each document in the Domino view, retrieves the appropriate field values and assigns them to the Salesforce object. During each loop it specifies that the Standard object is a “Contact” salesforce object.



## The Software Solution Experts

```
SObject[] records = new SObject[noofdocs];
int i=0;
while (doc != null) {
    tmpdoc = view.getNextDocument(doc);
    SObject contact = new SObject();
    // SObject a = new SObject();
    //a.setType("Account");

    contact.setType("Contact");
    //      contact.setField("FirstName", "Harry Dunnnnne Oct2");
    contact.setField("LastName", doc.getItemValueString("FullName"));
    contact.setField("MailingStreet", doc.getItemValueString("CompanyName"));
    contact.setField("MailingCity", doc.getItemValueString("Address2"));
    contact.setField("MailingState", doc.getItemValueString("Address3"));
    contact.setField("MailingCountry", doc.getItemValueString("Address4"));
    contact.setField("MailingPostalCode", doc.getItemValueString("PostCode"));
    contact.setField("EMail", doc.getItemValueString("Email"));
    contact.setField("LondonEvent__c", "Yes");
    doc.recycle();
    doc = tmpdoc;

    records[i] = contact;
    i=i+1;
}
```

Finally the line that writes the records back to salesforce is...

```
SaveResult[] saveResults = connection.create(records);
```

The same process can be applied to countless other objects etc.. Some of the issues not discussed here, and this list is not exhaustive include..

- Testing with Salesforce. With a large number of records you could meet your limits in read/writes, so that would have to be born in mind.
- Marking the Domino documents as processed
- Pulling the Username and Password/SecurityToken from an encrypted store.

We hope this is of some use.

Bluewave Technology Group