The SOAPbox User’s Guide

Application Documentation Version 1.3

THE SOCIAL FOUNDRY

November 9, 2012
Congratulations on your purchase of the SOAPbox!

This guide is intended for users of the system (i.e. administrators), and assumes the reader has some basic knowledge of SQL based systems and web services. It also assumes you have an installed and ready to use SOAPbox. If you do not have one, you can access our cloud-based demonstration SOAPbox located at the following URL. Please contact support for the login credentials and WSDL definitions.

https://demo.demosf4.com/TheSoapBox/login.aspx

For instructions on how to install and configure your own SOAPbox, please see the SOAPbox Installation Guide in the support section of our website.
Table of Contents:

SOAPBox basics-3
Access Keys configuration-9
SQL Server database configuration
  Database connections-14
  SQL operations-21
  Stored procedures-31
Oracle Database configuration
  Database connections-37
  SQL operations-39
  Stored procedures-49
SOAPbox error logs-55
User maintenance-56
How to test your services and WSDLs-57
A new installation of the SOAPbox has the following default credentials:

User: admin
Password: admin
Once logged in, there are two main sections. The first is the menu partition; the second is the detail partition of each menu selection.
**Home Page:**

This menu displays vital statistics on the services that are running. It displays whether or not the services are enabled and any invalid access logged by the SOAPbox.

**DBBrowse** is the service you can invoke to get a listing of all available SQL operations.

**DBExecute** is the service you invoke to actually perform a database operations.
Web Service Status Page:
This page shows the status of each web service. It allows you the option to turn one or more of the services off.
To toggle the status of a service, hit the ‘Edit’ button, and set the active status to ‘Y’ or ‘N’.
Hit ‘Update’ to save your changes.
The Web Service Access History Page:

This page shows any invalid attempts to access the services. You can filter the report by selecting the filter icon at the top. If you want a clean log, simply hit the 'Purge Table' button and it will clear the access log.
**Access Key Setting Page:**

This page contains the access keys that you give to systems that want to access your Modules. **What is a Module?** A Module is a SQL action that you define in the SOAPbox. It can be a select, insert, update, or delete statement. It can even be a stored procedure. Modules contain the operations you want to perform on your database. Access keys are granted to modules. This grant process ensures that keys granted to outside systems can only access the Modules that you have enabled for them.

By expanding the access key record (Select the ‘>’ to the left of the key), you can see all of the Modules that this key can access.
To create a new Key select the ‘+’ next to the ‘Add Access Key’ option.
Give the Access Key a name. Once you have entered a name for your Key, you will need to define a key value.
You can set your own value OR by hitting the ‘Generate System Key’ button the system can assign a hard to guess value.
Hit the ‘insert’ button and the Access Key is created. Once the Access Key is created, the next step is to grant access to the various Modules defined in the system.
To grant a key access to a Module, select the ‘+’ icon next to the ‘Add Module to Key’ label. **If you are just starting this process skip this step as there will be no Modules defined in the system.** Go to Page 13 in this document.
Select the Module you want the Key to access and hit ‘insert’. This Key now has access to the ‘testNorthwind’ Module.

**TIP:** You may want to have different keys for different systems that need to access the service. You may have a key for Salesforce.com, a key for a custom application, and yet another key for Microsoft Dynamics for example. If you are a cloud provider, you can provide an API for each one of your tenants that wish to access your services programatically.
This page intentionally left blank.
**SQL Server Database Settings Page:**

This page contains the MS SQL Server database connection information.

To create a new connection hit the ‘+’ next to the ‘Add MS SQL Server Database’ label.
Enter in your database credentials:

**Connection Name:** Give it a name that makes sense for you.

**DB Username:** This is the database username.

**DB Password:** This is the database password.

**Host Name:** This is the http path of the SQL Server Database.

**Database Name:** This is the database name you wish to access.

**Records Threshold:** This is a throttle on the service to prevent long running SQL operations. If the records threshold of the returned SQL select statement exceeds this value, the service will terminate. This is to ensure that the administrator can control the record limit per connection for performance purposes.
**Tip:** Ensure your SQL Server instance is accessible via http (default port 1433 must be open), and it can accept remote incomings TCP connections. Most SQL Server databases already have these settings set, but in some cases you may need to set these up.

To check to make sure your instance can accept remote connections go to the following:

SQL Server Configuration Manager: Ensure that TCP/IP is enabled.
Microsoft SQL Server Management Studio: Right click on the instance, select connections and ensure that 'Allow remote connections to this server'. is checked.
Also check that the SQL Server Browser Service is running.
These suggestions should cover the majority of your connection issues if you cannot connect to your SQL Server Database of choice. As always, consult your vendor directly should these quick fixes not alleviate your connection problems.
The Stored SQL Page:

Perquisites: A valid database connection entry and a valid access key.

This page is where you can create MS SQL operations (i.e. Modules) that will be exposed as a web service.

To add a new Module select the ‘+’ next to the ‘Add Stored SQL Module’ label.
Give the Module a name and a description that works well for you.

Enter in the SQL statement you wish to expose (you can expose select, updates, inserts and delete operations).

**Note:** For SQL Server the binding syntax you must use the `@` symbol.

Finally select the connection you wish to use and hit ‘Insert’.
If you are passing parameters to the database operation you will need to define these parameters by selecting the ‘>’ symbol to the left of the Module name. This brings up the child records (both Parameters and Access Keys). If you are not passing any parameters to the Module then you can skip this step.

Select the ‘+’ next to the ‘Add New Parameter’ label to create a new parameter.

**TIP:** Parameters protect against SQL injection. Defining parameterized SQL operations is considered best practices and is enforced by the SOAPbox.
Enter in the parameter name and hit the 'Insert' link. **The parameter names in your SQL statement must match the parameter names defined in the section displayed below.** (Note: you do not need to place the ‘@’ in the parameter name.)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>customerid</td>
<td>10</td>
</tr>
</tbody>
</table>

**TIP:** What is the sequence used for? The sequence number is used to tell the service to expect a parameter in a certain order. When called the service the system will send all of the parameters in a single string. The sequence tells the system what parameter to look for first etc...
Finally you will need to grant access to this service to one or more Access Keys. The process of adding Access Keys ensures that only certain Access Keys can perform authorized SQL operations.

To add an Access Key click on the ‘+’ next to the ‘Add Access Key’ label.

**Tip:** You can add as many Access Keys to a Module as you would like.
Select the Access Key from the list and hit the 'Insert' link.
Once saved, this Module (and its SQL operations) are now web service enabled. This module can only be accessed by the ‘salesforce_nas1’ Access Key.
To test your Module, click on the ‘Test Module’ link next to the module.
The Test Module Page:

The purpose of this page is to allow administrators to test their services from within the application without having to use a generic SOAP client each and every time they make a change. This page is available to test both SQL operations and stored Procedure calls.

In the Parameter String section, I pass parameters in the order defined in the Modules parameter definitions (remember the sequence number? This is what it is used for).

**Note:** Parameters string values must be separated by a ‘|’. This tells the system when one parameter value ends and another begins.

Click on the ‘Test Web Service’ button. If you get results, the Output Box will display data. This confirms the service is working.

You can also test the services using a standard SOAP client. See the ‘Testing the services’ section of this document for more details.
Shameless Marketing plug and Key product point:
Allowing multiple operations to be called from a single web service end point (WSDL) makes the SOAPbox a lot more efficient and flexible when defining database web services. A single application needs only one WSDL definition to perform multiple operations. If you need to make changes, the WSDL NEVER changes, only the output does. This enables organizations to create and modify database web services at the fraction of the time and cost vs. traditional integration methods. Most solutions require a different WSDL definition for each operation and changes to such operations require a change in the WSDL. This, in turn requires all of the systems using that WSDL to change their code. A major headache.

Note about Web Service Performance:
Web services by definition contain higher overhead than a native SQL operations in the database, web service performance will be impacted by the following attributes.

- RAM of the SOAPbox.
- CPU of the SOAPbox.
- Size of the XML document returned.
- Performance of the target database.
The Stored Procedure Page:
This page is where you reference the stored procedures you wish to expose as a web service. Unlike the SQL section, you simply need to reference the stored procedure name as opposed to writing the SQL code directly.
To create a new stored procedure Module, select the ‘+’ next to the ‘Add Stored Procedure Module’ label.

Enter in a name and description that works well for you.

In the ‘Value’ field enter in the name of the stored procedure you wish to call.

Select the database connection you wish to use for this stored procedure call.

Select ‘Insert’ to create the Module.

**Tip:** The SOAPbox does not create the stored procedure, but rather just references an already existing stored procedure.

In order for the stored procedure to run, the connection you have selected must have the proper database privileges.
Once the Module has been created, parameters need to be defined. Since this is a stored procedure, you will need to ensure the parameters from the SOAPbox match the parameters in the database. This includes whether or not the parameters are IN or OUT. The sequence of the parameters as must also match (This is an MS SQL Server requirement).

Select the ‘>’ to the left of the Module name to bring up the child options for this Module.

Select the ‘+’ next to the ‘Add New Parameter’ label.
Enter in the parameter name as it is defined in your stored procedure definition. The sequence will auto default, however you can adjust the sequence of the parameter if you add new parameters or change the order. Finally, define whether or not this parameter is an IN parameter or an OUT parameter.

Select ‘Insert’ to create the new parameter. Repeat until all parameters are defined.
The following below shows that the Module takes in the companyname and orderid. It then performs the operation, per the store procedure in the database, and returns an output in the 'result' OUT parameter. Testing the parameter can be done just like testing a SQL Module.
Adding Access Keys to a Module is the same as well.
Select the Access Key in the pick list.
Hit ‘Insert’. 
The Oracle Database Settings Page:

In this page you will define the Oracle connection settings to access an Oracle database.

Note: Oracle requires a separate installation of client access software. If you are connecting to an Oracle database, please see the Oracle ODAC Installation Guide. The SOAPbox hardware appliance will already have this installed, while the software version requires a separate installation.

Select the ‘+’ next to the ‘Add Oracle Database’ label to create a new connection.
**Connection Name:** Give it a name that makes sense for you.

**DB Username:** This is the database username.

**DB Password:** This is the database password.

**TNS/DSN:** This is TNS entry for the database connection.

Select the ‘Insert’ button.

**Note:** In the TNS/DSN section place the TNS connection information as seen below. This will allow you to bypass the manual creation of a system or user DSN on the machine. You can also create a system DSN on the machine and reference the DSN name here, however this is not covered in this manual.

**Tip:** Oracle TNS entries can be tricky. We recommend cutting and pasting an existing TNS entry v.s. trying to type one in from scratch.
The Stored SQL Page:

Perquisites: A valid database connection entry and a valid access key.

This page is where you can create Oracle operations (i.e. Modules) that will be exposed as a web service.

To add a new Module select the ‘+’ next to the ‘Add Stored SQL Module’ label.
Give the Module a name and a description that works well for you.

Enter in the SQL statement you wish to expose (you can expose select, updates, inserts and delete operations).

**Note:** For Oracle the binding syntax you must use is the `:` symbol.

Finally select the connection you wish to use and hit ‘Insert’.
If you are passing parameters to the database operation you will need to define these parameters by selecting the ‘>’ symbol to the left of the Module name. This brings up the child records (both Parameters and Access Keys). If you are not passing any parameters to the Module then you can skip this step.

Select the ‘+’ next to the ‘Add New Parameter’ label to create a new parameter.

**TIP:** Parameters protect against SQL injection. Defining parameterized SQL operations is considered best practices and is enforced by the SOAPbox.
Enter in the parameter name and hit the 'Insert' link. **The parameter names in your SQL statement must match the parameter names defined in the section displayed below.** (Note: you do not need to place the ‘:’ in the parameter name.)

**TIP:** What is the sequence used for? The sequence number is used to tell the service to expect a parameter in a certain order. When calling the service all of the parameters will be sent in a single string. The sequence tells the system what parameter to look for first etc...
Finally you will need to grant one or more Access Keys to the Module. The process of adding Access Keys ensures that only certain Access Keys can perform authorized SQL operations.

To add an Access Key click on the ‘+’ next to the ‘Add Access Key’ label.

**Tip:** You can add as many Access Keys to a Module as you would like.
Select the Access Key from the list and hit the ‘Insert’ link.
Once saved, this Module (and its SQL operations) are now accessible via web service calls. This module can only be accessed by the ‘salesforce_nas1’ Access Key.
To test your Module, click on the ‘Test Module’ link next to the Module.

![Oracle Stored SQL Modules]

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Description</th>
<th>Database Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor Invoice Listing</td>
<td>This will return a listing of invoices from a given vendor for a given date</td>
<td>Oracle Production Database</td>
</tr>
</tbody>
</table>

**Add New Parameter**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_creation_date</td>
<td>10</td>
</tr>
<tr>
<td>x_vendor_id</td>
<td>20</td>
</tr>
</tbody>
</table>

**Add Access Key**

Access Key

```
salesforce_nas1
```
The Test Module Page:

The purpose of this page is to allow administrators to test their services from within the application without having to use a generic SOAP client each and every time they make a change. This page is available to test both SQL operations and stored Procedure calls.

In the Parameter String section, you can pass parameters in the order defined in the Modules parameter definitions (remember the sequence number? This is what it is used for).

**Note:** Parameters string values must be separated by a ‘|’. This tells the system when one parameter value ends and another begins.

Click on the ‘Test Web Service’ button. If you get results, the Output Box will display data. This confirms the service is working.

You can also test the services using a standard SOAP client. See the ‘Testing the services’ section of this document for more details.
Shameless Marketing plug and Key product point:

Allowing multiple operations to be called from a single web service end point (WSDL) makes the SOAPbox a lot more efficient and flexible when defining database web services. A single application needs only one WSDL definition to perform multiple operations. If you need to make changes, the WSDL NEVER changes, only the output does. This enables organizations to create and modify database web services at the fraction of the time and cost vs. traditional integration methods. Most solutions require a different WSDL definition for each operation and changes to such operations require a change in the WSDL. This, in turn requires all of the systems using that WSDL to change their code. A major headache.

Note about Web Service Performance:

Web services by definition contain higher overhead than a native SQL operations in the database, web service performance will be impacted by the following attributes.

- RAM of the SOAPbox.
- CPU of the SOAPbox.
- Size of the XML document returned.
- Performance of the target database.
**The Stored Procedure Page:**

This page is where you define the stored procedures you wish to expose as a web service. Unlike the SQL section, you simply need to define the stored procedure name as opposed to writing the SQL directly.

**Tip:** For Oracle databases, you can use both stored procedures directly OR use packaged procedures as well. For stored procedures, just enter in the procedure name. For packaged procedures, enter in the package name and procedure name as follows: `package_name.procedure_name`.

![Image of Oracle Stored Procedure Modules](image_url)
To create a new stored procedure call, select the ‘+’ next to the ‘Add Stored Procedure Module’.
Enter in a name and description that works well for you.
In the ‘Value’ field enter in the name of the stored procedure you wish to call.
Select the database connection you wish to use for this stored procedure call.
Select ‘Insert’ to create the Module.

**Tip:** The SOAPbox does not create the stored procedure, but rather just references an already existing stored procedure.
In order for the stored procedure to run, the connection you have selected must have the proper database privileges.
Once the Module has been created, parameters need to be defined. Since this is a stored procedure, **you will need to ensure the parameters from the SOAPbox match the parameters in the database.** This includes whether or not the parameters are IN, OUT, or INOUT. You must also match the parameter sequences as well.

Select the ‘>’ to the left of the Module name to bring up the child options for this module.

Select the ‘+’ next to the Add New Parameter label.
Enter in the parameter name as it is defined in your target database. The sequence will auto default, however you can adjust the sequence of the parameter calls if you add new parameters or change the order. Finally, define whether or not this parameter is an IN parameter, OUT Parameter or an INOUT parameter.

Select ‘Insert’ to create the new parameters. Repeat until all parameters are defined.
The following below shows that the Module takes in the vendor_id, and new_terms parameter. It then performs the operation, per the store procedure in the database, and returns an output in the ‘x_status’ OUT parameter as well as returning the ‘x_new_term’s parameter value as well. Testing a stored procedure is the same as testing a SQL Module.
Adding Access Keys to a Module is the same as adding an Access Key to a SQL Module.

Select the Access Key in the pick list.

Hit the ‘Insert’ link.
The SOAPbox Error Log Page:

This page shows any SOAPbox errors that have occurred when the service is called.

You can purge the log by hitting the ‘Purge Table’ button.
The Define Users Page:

This page is where you can define additional SOAPbox users in order to maintain audit and compliance standards for your organization.

To add a new user select the ‘+’ next to the ‘Add User’ label.
Enter in your new username and password:
Select the ‘Insert’ link to create the user.

Note: The SOAPbox requires at least one uppercase, one number or special character, and it must be at least 8 characters long.
The Login History Page:

This page shows the login audit trail for each user as they attempt to log into the SOAPbox.

You can purge the log by hitting the ‘Purge Table’ button.
How to test your services and WSDLs.

The SOAPbox has two WSDLs.

The first is DBBrowsen: This WSDL can be found at the following URL.

https://<enter in your domain here>/TheSoapBox/WebServices/DBBrowse.asmx?wsdl

DBBrowsen will allow you to browse the listing of all defined Modules in your SOAPbox as well as the parameter information for each. Think of this as a registry of all possible actions your SOAPbox has defined.

You can use a SOAP client such as SOAPUI to import the WSDL in order to make a standard SOAP call.

Below is a sample SOAP call to the service.

I can pass the key (from the Access Key settings either Salesforce encrypted or not depending on your version) and then invoke the service.

SOAP Method: Browse

ServerAddress: [URL]

Browse.Key[0]: [Input field]
If the key is authenticated successfully the web service will return a well formed XML document you can utilize in your custom code.

```xml
</parameters>
</module>
▼<module>
 ▼<module_alias>Customer Orders</module_alias>
 ▼<module_type>SQL</module_type>
 ▼<description>
 This will return the orders for a customer when you pass the customerid and the companyname
 </description>
 ▼<database_name>MS SQL Production</database_name>
 ▼<database_type>SQLSERVER</database_type>
 ▼<parameters>
 ▼<parameter>
  ▼<sequence>10</sequence>
  ▼<parameter_name>customerid</parameter_name>
  ▼<parameter_direction>IN</parameter_direction>
 </parameter>
 ▼<parameter>
  ▼<sequence>20</sequence>
  ▼<parameter_name>companyname</parameter_name>
  ▼<parameter_direction>IN</parameter_direction>
 </parameter>
 ▼<parameters>
</module>
```


59
The second web service is DBExecute. This WSDL can be found at the following URL.

https://demo.demosf4.com/TheSoapBox/WebServices/DBExecute.asmx?wsdl

In order for this to execute successfully I need to pass:

- The Module name I want to call.
- The parameters:
- The Key.
If the key is authorized, I get a well formed XML document with the data requested. You can then use this SOAP response in your custom coding.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <ExecuteResponse xmlns="http://www.thesocialfoundry.com/">
      <ExecuteResult>
          <rows>
            <row>
              <VENDOR_NAME>GE Capital</VENDOR_NAME>
              <INVOICE_ID>193778</INVOICE_ID>
              <INVOICE_NUM>ERS-9128-147802</INVOICE_NUM>
            </row>
            <row>
              <VENDOR_NAME>GE Capital</VENDOR_NAME>
              <INVOICE_ID>193939</INVOICE_ID>
              <INVOICE_NUM>ERS-5158-147922</INVOICE_NUM>
            </row>
            <row>
              <VENDOR_NAME>GE Capital</VENDOR_NAME>
              <INVOICE_ID>194625</INVOICE_ID>
              <INVOICE_NUM>ERS-5176-148485</INVOICE_NUM>
            </row>
            <row>
              <VENDOR_NAME>GE Capital</VENDOR_NAME>
              <INVOICE_ID>194159</INVOICE_ID>
              <INVOICE_NUM>ERS-9492-148068</INVOICE_NUM>
            </row>
          </rows>
        </Elements>
      </ExecuteResult>
    </ExecuteResponse>
  </soap:Body>
</soap:Envelope>
```
END