

# Accessing MySQL from Ruby

---

## Contents

1. Introduction .....	2
2. Prerequisites .....	2
3. Setup .....	2
3.1. Populate MySQL Database.....	2
3.2. Set up ODBC Access .....	3
4. Discovery.....	4
4.1. Web Service Creation using SOA Gateway .....	4
4.2. Accessing the WSDL .....	6
5. Accessing a Web Service with Ruby.....	8
Ruby Editor.....	8
Ruby Code .....	8
Debugging Code .....	9
6. Conclusion.....	10
7. Appendix .....	10

## 1. Introduction

In this tutorial we will show you how to build a Ruby application to access MySQL via the SOA Gateway.

## 2. Prerequisites

It is assumed that you are running the 3 components, MySQL, Ruby and the SOA Gateway on Windows.

It is assumed you already have a SOA Gateway server and Control Centre installed. See [here](#) for more info about installing the SOA Gateway.

## 3. Setup

Download the latest binaries for Ruby from the [Ruby downloads page](#), install it according to the install.txt document in the Ruby distribution library. For this tutorial I downloaded Ruby 1.8.6 One-Click Installer and followed the default options.

You will also need a MySQL database. Again, the Open Source version (known as the *MySQL Community Server*) can be freely downloaded from the MySQL website. See [this link](#) for download, and [here](#) to step you through the installation and configuration.

### 3.1. Populate MySQL Database

Now that you've got MySQL installed and configured, you will need to populate it with some demo data. For this we use the RisarisBank sample. This is available [here](#).

- Save this file to "C:\Temp\RisarisBank.sql"
- Connect to the MySQL Server using the **mysql** command.

E.g *shell> mysql -u root -p*

This command connects to the server using the MySQL `root` account to make sure that you'll have permission to create the `RisarisBank` database. The `-p` option tells **mysql** to prompt you for the `root` password. Enter the password when prompted. (Remember that the MySQL `root` account is not the same as the operating system `root` account and probably will have a different password.)

- Create the `RisarisBank` database.

*mysql> CREATE DATABASE RisarisBank;*

*mysql> use RisarisBank;*

- Load the contents of `RisarisBank.sql` into the `RisarisBank` database. E.g.

*mysql> SOURCE c:\Temp\RisarisBank.sql*

- After the `SOURCE` command finishes, you can view your new tables.

```
mysql> SHOW TABLES;

mysql> DESCRIBE CustomerInformation;

mysql> DESCRIBE Branch;

etc ...
```

### 3.2.Set up ODBC Access

The final thing to do with your MySQL Database is to set up an ODBC DSN which will be used by the SOA Gateway to access this database.

Click Start, Control Panel, Administrative Tools, Data Sources (ODBC)

From the resulting screen, choose the “System DSN” Tab.

Click Add

From the list of data source drivers, select “MySQL ODBC 3.51 Driver”.

*If you do not see this driver in the list, you need to install the MySQL Connector. See [here](#) for more information. We recommend installing v3.51.*

Click Finish, and a window will appear allowing you to enter the DSN information. Add the following:

Data Source Name: RisarisBank

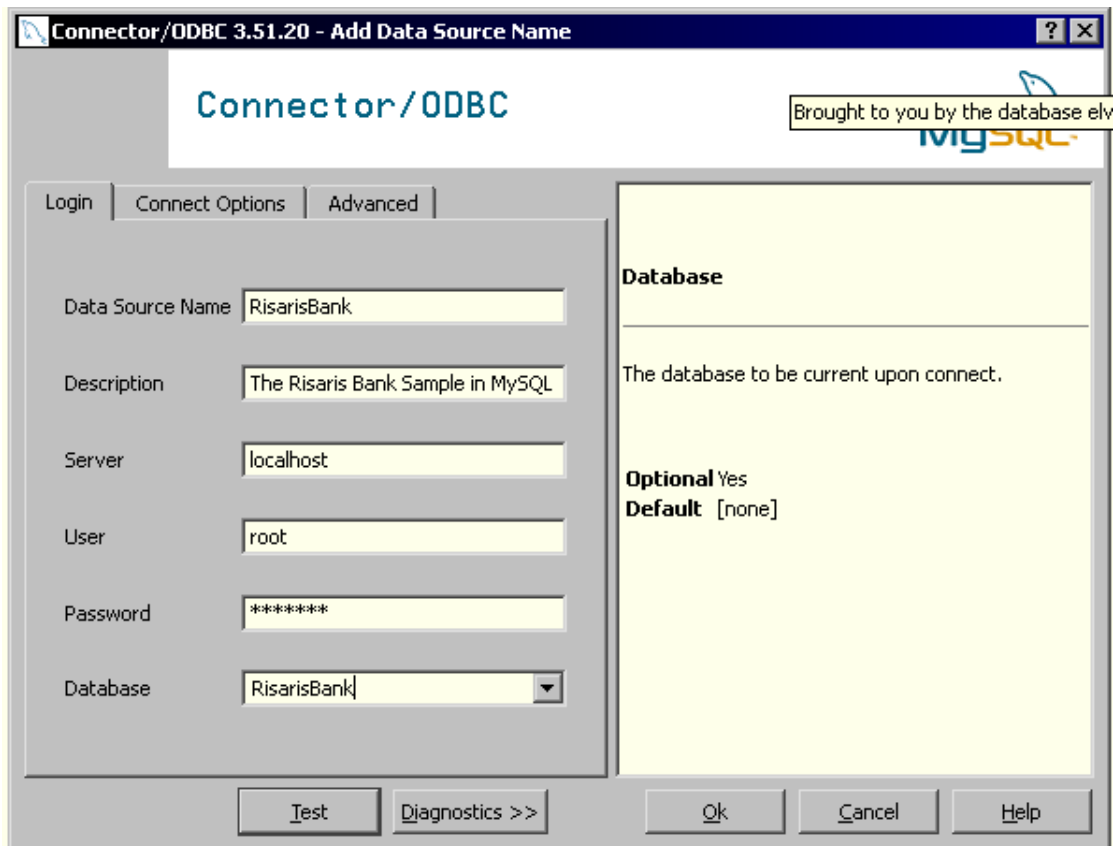
Description: The Risaris Bank Sample in MySQL

Server: localhost

User: root

Password: \*\*\* your MySQL root password \*\*\*

Database: RisarisBank (select from the drop down list)



All other options can be left as-is. Click OK.

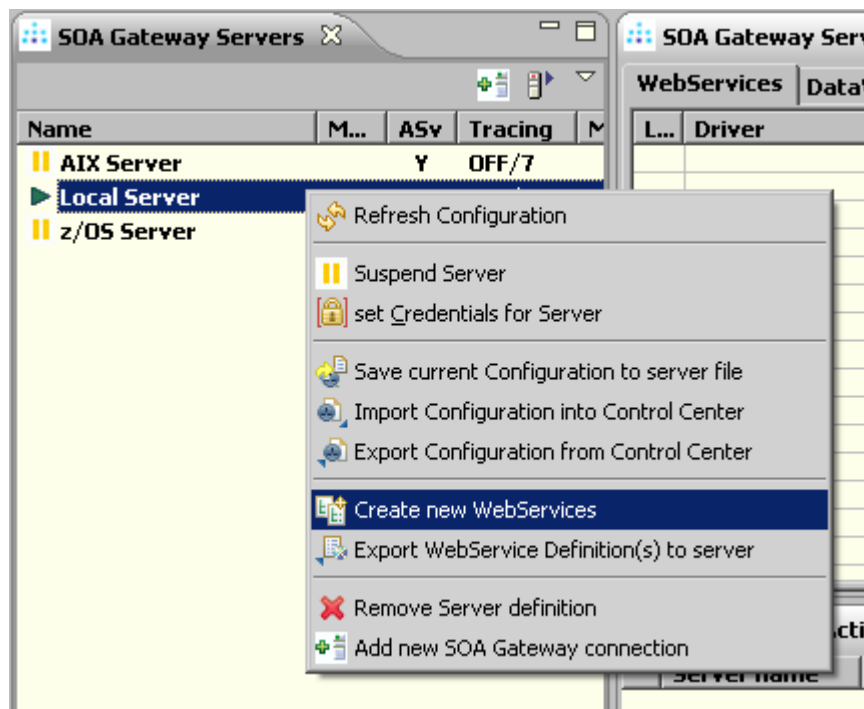
## 4. Discovery

At this stage you've got Ruby installed, and a MySQL database with some sample data in it. In this section we'll show you how to create web services from each of the MySQL tables. These web services can be used by the Ruby language (and many others) to give you direct real-time access to your MySQL Data.

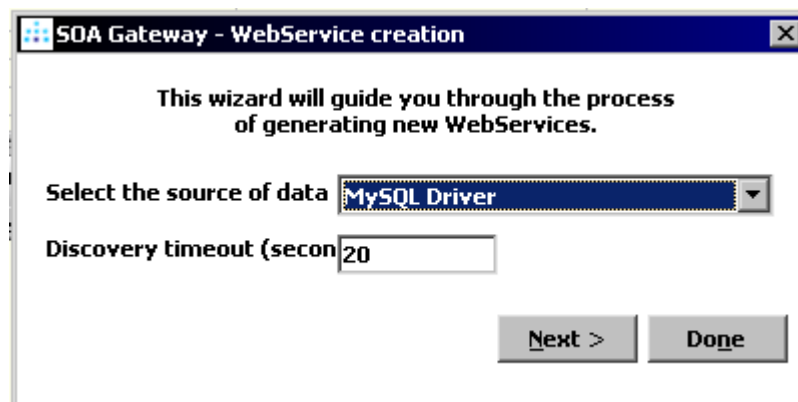
### 4.1. Web Service Creation using SOA Gateway

Start your SOA Gateway Control Centre. See [here](#) for an introduction to the Control Centre.

In your servers view, right click the entry which represents your local SOA Gateway Server. Select "Create New Web Services".



From the next dialog, choose “MySQL Driver”. If you do not see have a MySQL Driver in the list, see how to create one [here](#).



Click Next.

The next screen gives you the ability to add information about your DSN

**SOA Gateway WebService Creation (mysql)**

Step 1 - select input

Specify ODBC descriptors & option(s)

ODBC Dsn:

UserId / Password:

Pattern:

Max.entries to list:

Options for generated WebService name

Advanced Options: ☐

Discover Cancel

Enter the above information and click Discover.

The wizard will display all the tables it finds at this (RisarisBank) DSN.

Click "Select All", and click "Import".

The wizard will create web services from each one of these tables.

Name	M...	ASv	Mod	Driver	WebService	DataSource Id	DataView
AIX Server		Y	MySQL	MySQL Driver	accountsmovements	odbcDsn=RisarisBank, tableName=accountsmovements	accountsmovements
DM2		Y	MySQL	MySQL Driver	audit	odbcDsn=RisarisBank, tableName=audit	audit
dublin dev		Y	MySQL	MySQL Driver	branch	odbcDsn=RisarisBank, tableName=branch	branch
jk server		Y	MySQL	MySQL Driver	currentaccount	odbcDsn=RisarisBank, tableName=currentaccount	currentaccount
jk server linux		Y	MySQL	MySQL Driver	customeraccountxref	odbcDsn=RisarisBank, tableName=customeraccountxref	customeraccountxref
jom server		Y	MySQL	MySQL Driver	customerinformation	odbcDsn=RisarisBank, tableName=customerinformation	customerinformation
Local Server		Y	MySQL	MySQL Driver	depositaccount	odbcDsn=RisarisBank, tableName=depositaccount	depositaccount
lxbre server		Y	MySQL	MySQL Driver	tellertable	odbcDsn=RisarisBank, tableName=tellertable	tellertable
PCRJW9		Y					
risaris.com server		Y					
vse		Y					
z/OS Server		Y					
z/vse		Y					

You've just created 8 Web Services from your 8 MySQL Tables!

## 4.2.Accessing the WSDL

Web Service Description Language (WSDL) is a standard, XML-based language that is used to describe a Web Service.

For each of the 8 web services you've created in the previous section, the SOA Gateway provides you with a WSDL to describe the Web Service. The WSDL itself is usually interpreted by a web

service client, such as Ruby, but it is useful to know where to find the WSDL for each of your Web Services.

As WSDL is XML-based, it will open in your browser of choice. To see the WSDL for one of your Risaris Bank web services, do the following in your SOA Gateway Control Centre:

- Click on the web service you are interested in, for example the “branch” web service.
- The properties for this web service should appear in your [Properties View](#). If you do not see the Properties view, select Window -> Show View -> Other -> General -> Properties and click OK.
- In the properties view, there is a link to your WSDL. Click it to open the WSDL in a browser.

The screenshot displays the SOA Gateway Server Configuration interface for a local server. It features a table of web services and a detailed properties view for the 'branch' service.

Mod	Driver	WebService	DataSource Id
MySQL	MySQL Driver	accountsmovements	odbcDsn=RisarisBank, tableN
MySQL	MySQL Driver	audit	odbcDsn=RisarisBank, tableN
MySQL	MySQL Driver	branch	odbcDsn=RisarisBank, tableN
MySQL	MySQL Driver	currentaccount	odbcDsn=RisarisBank, tableN
MySQL	MySQL Driver	customeraccountxref	odbcDsn=RisarisBank, tableN
MySQL	MySQL Driver	customerinformation	odbcDsn=RisarisBank, tableN
MySQL	MySQL Driver	depositaccount	odbcDsn=RisarisBank, tableN
MySQL	MySQL Driver	tellertable	odbcDsn=RisarisBank, tableN

Below the table is the SOA Gateway Action Log, showing messages such as "ODBC discovery completed, 8 WebService(s) generated" and "Configuration autosaved due to published WebService modification(s)".

The Properties view for the 'branch' service is shown below the log. It includes fields for Name, DataView, Driver (MySQL), Read-only (unchecked), SBCS-Codepage, and MBCS-Codepage. The WSDL URL is displayed as <http://localhost:56000/branch?WSDL>. A green arrow points to the WSDL URL field.

Below the WSDL URL is the WebService Identification and options section, which includes fields for odbcDsn (RisarisBank), schemaName, and tableName (branch).

You can view the WSDL for the other web services by clicking the link from their properties view.

This WSDL is the starting point for using Web Services, and can be used time and again by different web service clients.

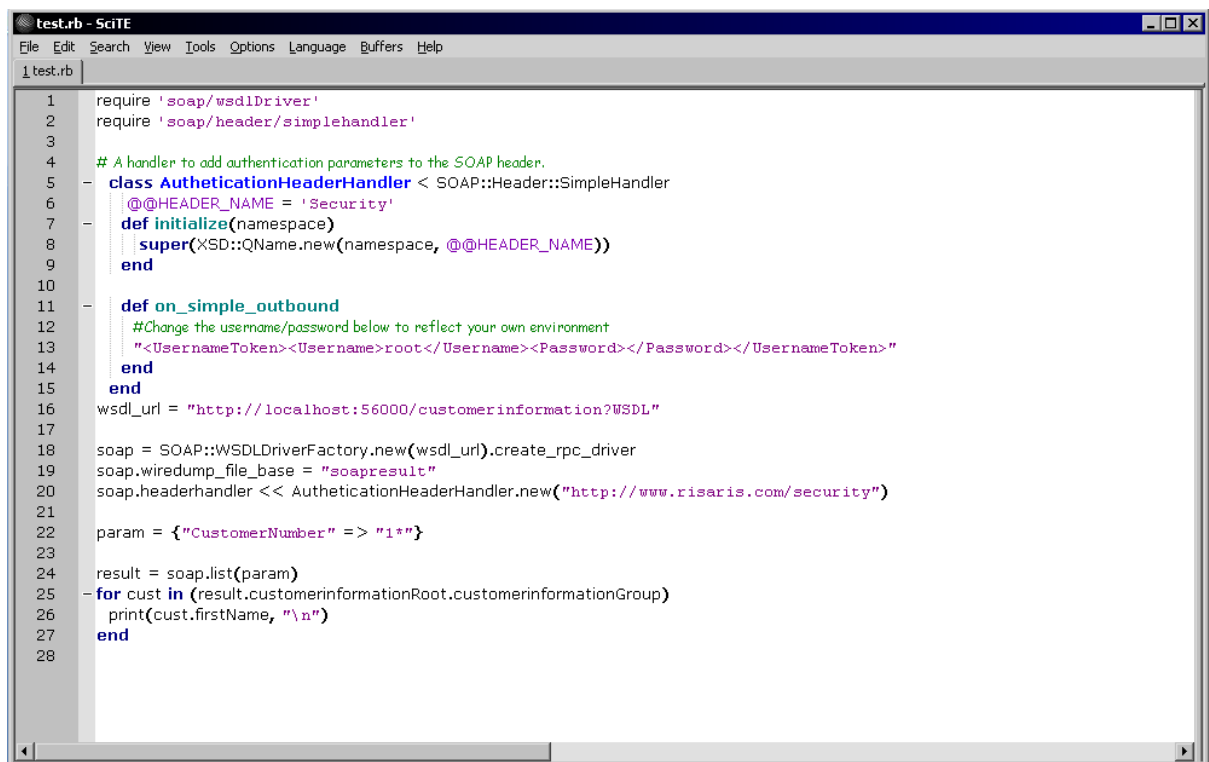
## 5. Accessing a Web Service with Ruby

We will use a Ruby script to access our new Risaris Bank Web Services via the WSDL. Those familiar with Ruby will probably want to change the script provided but it can be run with minimum changes described below. The whole script is provided in APPENDIX A.

### Ruby Editor

Use the editor provided, SciTE.exe, which is located in folder scite where you installed Ruby. Copy the code from Appendix A and save the code as the filename RBCustomerTutorial.rb (as in example below).

Hit F5 and the results will appear in the right-hand pane:

The image shows a screenshot of the SciTE Ruby editor window. The title bar reads "test.rb - SciTE". The menu bar includes File, Edit, Search, View, Tools, Options, Language, Buffers, and Help. The editor contains a Ruby script for testing a web service. The script starts with two require statements for 'soap/wsdlDriver' and 'soap/header/simplehandler'. It then defines a class 'AuthenticationHeaderHandler' that inherits from 'SOAP::Header::SimpleHandler'. The class has an '@@HEADER\_NAME' attribute set to 'Security' and an 'initialize' method that calls 'super' with 'XSD::QName.new(namespace, @@HEADER\_NAME)'. There is also an 'on\_simple\_outbound' method that constructs a SOAP header string with 'UsernameToken', 'Username' (root), and 'Password'. The script sets 'wsdl\_url' to 'http://localhost:56000/customerinformation?WSDL', creates a 'SOAP::WSDLDriverFactory' object, and sets 'soap.wiredump\_file\_base' to 'soapresult'. It then creates an 'AuthenticationHeaderHandler' object and sets a parameter 'CustomerNumber' to '1\*'. Finally, it calls 'soap.list' and iterates over the results to print the first name. The script ends with 'end' on line 28.

```
1 require 'soap/wsdlDriver'
2 require 'soap/header/simplehandler'
3
4 # A handler to add authentication parameters to the SOAP header.
5 class AuthenticationHeaderHandler < SOAP::Header::SimpleHandler
6   @@HEADER_NAME = 'Security'
7   def initialize(namespace)
8     super(XSD::QName.new(namespace, @@HEADER_NAME))
9   end
10
11   def on_simple_outbound
12     #Change the username/password below to reflect your own environment
13     "<UsernameToken><Username>root</Username><Password></Password></UsernameToken>"
14   end
15 end
16 wsdl_url = "http://localhost:56000/customerinformation?WSDL"
17
18 soap = SOAP::WSDLDriverFactory.new(wsdl_url).create_rpc_driver
19 soap.wiredump_file_base = "soapresult"
20 soap.headerhandler << AuthenticationHeaderHandler.new("http://www.risaris.com/security")
21
22 param = {"CustomerNumber" => "1*"}
23
24 result = soap.list(param)
25 for cust in (result.customerinformationRoot.customerinformationGroup)
26   print(cust.firstName, "\n")
27 end
28
```

### Ruby Code

Depending on your MySQL settings you will have to change the username/password provided in the on\_simple\_outbound method. In this example the user is root with no password.

"<UsernameToken><Username>root</Username><Password></Password></UsernameToken>"



Note also the WSDL URL <http://localhost:56000/customerinformation?WSDL>. Again depending on your SOA Gateway configuration you may need to change the server/port number.

In this example provided we are issuing a **list** request on the customer information table in the RisarisBank database. The list request can handle a wildcarded customer number key:

```
param = {"CustomerNumber" => "1*"}
```

```
result = soap.list(param)
```

Finally we can query the result for the customer's first name:

```
for cust in (result.customerinformationRoot.customerinformationGroup)
```

```
  print(cust.firstName, "\n")
```

```
end
```

Obviously this can be replaced by any of the column names in the customer information table.

## Debugging Code

The code provided should work off the bat. However, depending on your MySQL configuration, the RisarisBank tables may not be in lowercase as in our example e.g. customerinformation versus CustomerInformation. If this is the case the code will not work as is and all occurrences of customerinformation should be replaced by CustomerInformation.

The `p` and `puts` functions are useful for checking the content of variables. For example `p result` after the `result = soap.list(param)` call gives the following information and is useful for checking column names for adding new code.

```
#<SOAP::Mapping::Object:0x3240c04
{}customerinformationRoot=#<SOAP::Mapping::Object:0x3240ad8
{}customerinformationGroup=[#<SOAP::Mapping::Object:0x32409ac
{}CustomerNumber="1" {}FirstName="Casper" {}Surname="Ankergren"
{}AddressLine1="34 Green Street" {}AddressLine2="Crosses Street"
{}City="Mansfield" {}Postcode="MA5 9AJ" {}DateOfBirth="30/05/1978">,
#<SOAP::Mapping::Object:0x323f4bc {}CustomerNumber="10"
{}FirstName="Grenville" {}Surname="Dekker" {}AddressLine1="22 Uttaxter
New Road" {}AddressLine2="Chaddesden" {}City="Leeds" {}Postcode="LS4
9WB" {}DateOfBirth="21/10/1974">, #<SOAP::Mapping::Object:0x323e0b2
{}CustomerNumber="11" {}FirstName="Jane" {}Surname="Freeman"
{}AddressLine1="3 Portland Street" {}AddressLine2="Lichfield"
{}City="Gloucester" {}Postcode="GL4 8KD" {}DateOfBirth="26/10/1980">,
#<SOAP::Mapping::Object:0x323ccb2 {}CustomerNumber="12"
{}FirstName="Dale" {}Surname="Rolling" {}AddressLine1="The Hunters Rest"
{}AddressLine2="Branston" {}City="Preston" {}Postcode="PR4 4HG"
{}DateOfBirth="06/12/1989">, #<SOAP::Mapping::Object:0x323b8b2
{}CustomerNumber="13" {}FirstName="Arnold" {}Surname="Corrigan"
{}AddressLine1="14 Anderson Street" {}AddressLine2="Alleestree"
{}City="Derby" {}Postcode="DE6 8FT" {}DateOfBirth="10/06/1970">,
#<SOAP::Mapping::Object:0x323a4b2 {}CustomerNumber="14"
{}FirstName="Alfred" {}Surname="Summerscale" {}AddressLine1="3701 S.
```

```

George Mason" {}AddressLine2="Melbourne" {}City="Bath" {}Postcode="BA6
8UU" {}DateOfBirth="07/08/1982">, #<SOAP::Mapping::Object:0x32390b2
{}CustomerNumber="15" {}FirstName="Peter" {}Surname="Spiegel"
{}AddressLine1="11663 Charter Oak Co" {}AddressLine2="Foremark"
{}City="Manchester" {}Postcode="M98 4GT" {}DateOfBirth="29/12/1971">,
#<SOAP::Mapping::Object:0x3237cb2 {}CustomerNumber="16"
{}FirstName="John" {}Surname="Sviridov" {}AddressLine1="3319 Rosemere
Court" {}AddressLine2="Brailsford" {}City="Leeds" {}Postcode="LS8 5QQ"
{}DateOfBirth="12/06/1984">, #<SOAP::Mapping::Object:0x32368b2
{}CustomerNumber="17" {}FirstName="Sarah" {}Surname="Hall"
{}AddressLine1="4 Denmark Lane" {}AddressLine2="East Bridgford"
{}City="Portsmouth" {}Postcode="PO9 7QN" {}DateOfBirth="12/02/1983">,
#<SOAP::Mapping::Object:0x32354b2 {}CustomerNumber="18"
{}FirstName="Elspeth" {}Surname="Jones" {}AddressLine1="12375 W. Ohio
Circle" {}AddressLine2="Risley" {}City="Birmingham" {}Postcode="B93 1IO"
{}DateOfBirth="28/08/1986">, #<SOAP::Mapping::Object:0x32340b2
{}CustomerNumber="19" {}FirstName="Nigel" {}Surname="Wyllis"
{}AddressLine1="14 Borough Road" {}AddressLine2="Ockbrook"
{}City="Leeds" {}Postcode="LS8 2WL" {}DateOfBirth="29/04/1981">]>>

```

## 6. Conclusion

This tutorial shows how to access MySQL from Ruby using the SOA Gateway. As you can see, you have built a powerful application that uses Web Services to retrieve information in real-time.

## 7. Appendix

```

require 'soap/wsdlDriver'

require 'soap/header/simplehandler'

# A handler to add authentication parameters to the SOAP header.

class AutheticationHeaderHandler < SOAP::Header::SimpleHandler

  @@HEADER_NAME = 'Security'

  def initialize(namespace)

    super(XSD::QName.new(namespace, @@HEADER_NAME))

  end

  def on_simple_outbound

    #Change the username/password below to reflect your own environment

    "<UsernameToken><Username>root</Username><Password></Password></UsernameTok
en>"

```

```

        end

    end

    wsdl_url = "http://localhost:56000/customerinformation?WSDL"

    soap = SOAP::WSDLDriverFactory.new(wsdl_url).create_rpc_driver

    soap.wiredump_file_base = "soapresult"

    soap.headerhandler <<
    AutheticationHeaderHandler.new("http://www.risaris.com/security")

    param = {"CustomerNumber" => "1*"}

    result = soap.list(param)

    for cust in (result.customerinformationRoot.customerinformationGroup)
        print(cust.firstName, "\n")
    end
end

```