

RemoteQuery for iPhone and iPod Touch

Installation and configuration of the server-side components

Table of content

Table of content	1
Introduction.....	2
How to set up RemoteQuery.....	2
Disclaimer	2
Step 1: Get the RemoteQuery server-side components	3
Where to get the server side components.....	3
Description of the server-side components	3
Microsoft platform	3
Java platform	3
Step 2: Installation of the server-side components	5
Installation with IIS	5
Installation in a servlet container (e.g. Apache/Tomcat)	5
Step 3: Configuring the database connections.....	7
The configuration.xml file.....	7
Definition of the databases connections.....	7
Sample database connections for Windows / IIS.....	8
Sample database connections for Java	10
Special cases	12
Step 4: Configuring the user authorizations.....	13
Step 5: Validating the configuration.xml file	14
Validating the configuration file (Windows)	14
Validating the configuration file (Java).....	14
Step 6: Configuring the iPhone/iPod clients.....	15
Setting the connection(s) to RemoteQuery server(s)	15
Getting the device id (UDID).....	15
Miscellaneous.....	16
Tracing (Windows).....	16
Tracing (Java)	16

Introduction

RemoteQuery is an iPhone /iPod application that allows the user to remotely browse any relational database through a network (cellular or WiFi) connection, provided that the corresponding server-side components are installed and properly configured on one or more web servers “visible” on the network (see below for more details).

RemoteQuery communicates to the server side components via HTTP requests; it may be configured (see the RemoteQuery User Guide) for connecting to one or many servers, each one being the gateway to a different set of databases.

The server side components are the “web pages” or “servlets” that responds to the call from RemoteQuery, retrieving data from databases, formatting them and returning them to the iPhone/iPod. Those components are therefore hosted in a web site; and the web site and components must be visible over the network connection: that is, if you want to access to data from the cellular network, the web site must be visible as an “internet” URL. Otherwise, you may use one or more web site only visible in an “intranet” network, via WiFi.

How to set up RemoteQuery

Setting up RemoteQuery is a simple process, consisting of the following steps:

- 1) get the server-side components of RemoteQuery
- 2) install them on one or more web servers
- 3) configure your database connections
- 4) configure your users
- 5) validate the configuration
- 6) configure the connections on the users’ mobile phones

Each step is described in detail in the next chapters.

Disclaimer

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Step 1: Get the RemoteQuery server-side components

Where to get the server side components

The server side components can be retrieved, at no cost, in both source and executable format, at the producer web site (<http://www.logicainformatica.it/remotquery/server>). They are available for two platforms:

- Microsoft platform: (classic asp/vbscript), for an IIS web server
- Java platform: for an Apache Tomcat (or equivalent) servlet container

You may mix both technologies on the same server or on different servers.

Description of the server-side components

Microsoft platform

The server side components for the Microsoft IIS web server (packed into the “RemoteQuery MS.zip” file) are:

File name	Description
configuration.xml	Sample configuration file
configuration.xsd	XML schema of the configuration file, used by the RemoteQuery.asp page and by the validate.vbs script for XML validation
RemoteQuery.asp	web page (vbscript asp page) that receives and executes the requests coming from the iPhone application via HTTP
validate.vbs	script for verifying all syntax and content of the configuration.xml file

All file should be unzipped to the same folder (see next chapter).

Java platform

The web application RemoteQuery (packed into the “RemoteQuery.war” file) for the Apache Tomcat web server has the following structure:

File name	Description
default.htm	The empty default page of the web application
WEB-INF	
lib	Application libraries
log4j-1.2.4.jar	
jdom.jar	
commons-codec-1.3.jar	
classes	Application code
RemoteQuery.java	Servlet source java code
RemoteQuery.class	
log4j.properties	Properties file for Apache Log4j. This tool is used to log on file the servlet activity. You can edit this file to activate/deactivate logging and set the path of the log file. See: http://logging.apache.org/log4j/1.2/index.html

web.xml	Web application configuration file
configuration.xml	Sample configuration file
configuration.xsd	XML schema of the configuration file, used by the RemoteQuery.asp page and by the validate.vbs script for XML validation
META-INF	
MANIFEST.MF	

Step 2: Installation of the server-side components

The server-side components are made to be installed in a web server, that must be able to connect to the database(s) you want to make available to your iPhone/iPod users. Therefore, a good place for them is a server that hosts applications that currently use those databases.

Installation with IIS

In a Microsoft server, three files (`remoteQuery.asp`, `configuration.xml` and `configuration.xsd`) need to be installed in a web site or virtual directory under any IIS site with **anonymous login** or **basic authentication** allowed; in the latter case, `userid` and `password` for accessing the site will be specified when configuring each iPhone client.

Note that the access to databases is done in the `.asp` page, using the **connection strings** specified in the `configuration.xml` files; thus, the Windows authorization must be set in such a way that the executing user is authorized to use the proper interface, and the server must have the proper DB client library installed. For example, for accessing **Oracle**, the Oracle Client must be installed and the Oracle Ole DB Provider (or the equivalent Microsoft Oracle Provider) too; moreover the executing user (either the default IIS user or the user specified by the connected client) must usually have read-authority on the Oracle "bin" folder.

The site may also be protected with **SSL** encryption: this means that, on server side, a digital certificate will be applied to the site, and the client will call using the **https** instead of the **http** protocol; this distinction is reflected in the way you configure the iPhone clients.

In summary, the RemoteQuery prerequisites are those of [a basic web site running classic ASP pages](#).

Installation in a servlet container (e.g. Apache/Tomcat)

The RemoteQuery servlet comes packaged as a standard Java web application, `RemoteQuery.war`, which can be easily deployed in any compatible servlet container. The most commonly used server is Apache Tomcat and the rest of this section describes how to install the application on a Tomcat server.

- Stop Tomcat, if it is running.
- Remove these files and directories if they exist:
 - `TOMCAT_HOME/webapps/RemoteQuery.war`
 - `TOMCAT_HOME/webapps/RemoteQuery`
 - `TOMCAT_HOME/work`
- Copy the file `RemoteQuery.war` to `TOMCAT_HOME/webapps`.
- Start Tomcat.

You should install in the server `lib` folder of Tomcat all jdbc driver libraries need for your databases. Some of these libraries are, for example, (use the library version compatible with your DBMS):

<code>ojdbc14.jar</code>	Oracle jdbc driver
<code>postgresql-8.4-701.jdbc3.jar</code>	PostgreSQL 8.4
<code>mysql-connector-java-5.1.7-bin.jar</code>	MySQL 5.1.7
<code>sqljdbc.jar</code>	Microsoft SQLServer

After Tomcat starts, the war content is expanded in the 'TOMCAT_HOME/webapps/RemoteQuery' folder with the structure described above.

To modify the configuration files (configuration.xml, log4j.properties, or web.xml) you should stop the web application and restart it when changes have been done.

Step 3: Configuring the database connections

The configuration.xml file

The **configuration.xml** file is a file in XML format, following the syntax described in the associated XML schema file (**configuration.xsd**). The configuration file defines both the “logical databases” and the users of RemoteQuery. The configuration file can be edited and modified with any text editor, but an XML editor is suggested, if available (for instance, XMLSpy, Visual Studio or Eclipse).

After editing it, its format and syntax should be verified by using the validation utility (see “Step 5: Validating the configuration”).

NOTE: *the server-side package is supplied with a sample configuration file, that must be replaced by yours.*

Definition of the databases connections

A “logical database” is an XML element that defines a connection to a physical database and, possibly, a restriction to a given “schema” and/or to a specific subset of the entire database. Note that the user definition (see next section) allows you to restrict the access of each user to a different subset of all listed databases.

NOTE: *RemoteQuery can be used for granting wide range access to company’s databases to people like DBA, developers, operators, but it can also be used to provide very specialized views and tables to business users. The distinction can be made by properly configuring the logical databases, ranging from a simple full set of all tables and views, to a very customized subset of business views.*

Logical databases are defined as “database” tags in the “databases” group, as shown in the following samples:

```
<?xml version="1.0" standalone="yes" ?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="configuration.xsd" >
  <!-- List of databases -->
  <databases>

    <database name="Demo: IpseDixit" schema="" type="MSAccess"
      connectionstring="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\web\publi
c\RemoteQuery\IpseDixit EN.mdb;"/>

    <database name="Demo: Northwind" schema="northwind" type="MySQL"
      connectionstring="DRIVER={MySQL Connector/ODBC v5};SERVER=localhost;DATABAS
E=northwind;UID=RemoteQuery;PWD=RemoteQuery;OPTION=3"/>

    <database name="Demo: Scott" schema="SCOTT" type="Oracle"
      connectionstring="Provider=OraOLEDB.Oracle;Data Source=TEST;User ID=REMOTEQ
UERY;Password=REMOTEQQUERY"/>

  </databases>
</configuration>
```

Configuration file on Microsoft IIS

```
<?xml version="1.0" standalone="yes" ?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="configuration.xsd" >
  <!-- List of databases -->
  <databases>

    <database name="Demo: Northwind" schema="northwind" type="MySQL"
      driver="com.mysql.jdbc.Driver"
      connectionString="jdbc:mysql://localhost:3306/northwind?user=RemoteQuery&am
p;password=RemoteQuery" />

    <database name="Demo: Scott" schema="SCOTT" type="Oracle"
      driver="oracle.jdbc.driver.OracleDriver"
      connectionString="jdbc:oracle:thin:REMOTEQUERY/REMOTEQUERY@localhost:1521:T
EST"

  </databases>
</configuration>
```

Configuration file on Apache Tomcat web server

Here is the meaning and usage of the attributes of the “database” tag:

Attribute	Mandatory	Description
name	yes	name of the database as it will appear on the iPhone; it must be unique in the list, but the same name can be used on different configuration files
schema	no	schema name: if not empty, only table and views contained in the given schema will be shown
type	yes	Database type; can be one of the following: <ul style="list-style-type: none"> • MSAccess • Oracle • SQLServer • MySQL • PostgreSQL • SQLite^(new) • Other When “Other” is specified, the “query” attribute is mandatory
driver	Only for the Java platform	Is the JDBC driver to use for the connection
connectionstring	yes	connection string for the ADODB interface (Windows) or the JDBC interface (Java). See below for further explanation.
query	no	mandatory if type='Other'; see below for details.
asklogin	no	if =true, it means that the user will be requested a userid and password when accessing that DB (once per session). In this case, the connection string must contain the placeholders %user and %password (see below for more details).
updatable ^(new)	no	It must be = “true” for allowing update, insert and delete operations on the given database; this feature is available only if both the client (iPhone) application and the server components are version 2.0 or more.

Sample database connections for Windows / IIS

Here are some **examples** of database definition in Windows environment:

MSDAccess database

Note that no schema name is needed:

```
<database name="Demo: IpseDixit" schema="" type="MSAccess"
  connectionstring="Provider=Microsoft.Jet.OLEDB.4.0;
  Data source=C:\web\public\RemoteQuery\IpseDixit EN.mdb;"/>
```

MySQL database

In the following example, the connection is done using the ODBC adapter provided with the official MySQL distribution (other OLEDB provider exist).

```
<database name="Demo: Northwind"
  schema="northwind"
  type="MySQL"
  connectionstring="DRIVER={MySQL Connector/ODBC v5};
  SERVER=localhost;DATABASE=northwind;UID=RemoteQuery;PWD=RemoteQuery;OPTION=3"/>
```

Oracle database

In the following example, the connection is done using the Oracle OLE DB Provider, the Microsoft provider (MSDAORA) may be used as well.

```
<database name="Demo: Scott" schema="SCOTT" type="Oracle"
  connectionstring="Provider=OraOLEDB.Oracle;Data Source=TESTDB;
  User ID=REMOTEQUERY;Password=REMOTEQUERY"/>
```

Postgres database

```
<database name="RemoteQueryTest" schema="public" type="PostgreSQL"
  connectionstring="Provider=PostgreSQL OLE DB Provider;
  Password=xxxxxxx;
  User ID=postgres;
  Data Source=localhost;
  Location='RemoteQueryTest';Extended Properties='' />
```

SQLite database^(new)

```
<database name="RemoteQuery.sqlite" type="SQLite"
  connectionstring="Provider=OleSQLite.SQLiteSource.1;
  Data Source=C:\Progetti\iPhone\RemoteQuery\RemoteQuery.sqlite" updatable="true"/>
```

SQLServer database

In this example, userid and password are specified with the special placeholders **%user** and **%password** from the connection string, and the `asklogin=true` attribute is specified: this means that each user will be requested with a userid and password at first access (the same can be done with any database type).

```
<database name="sqltest" schema="myschema" type="SQLServer"
  connectionstring="Provider=SQLOLEDB.1;Initial Catalog=myschema;Data
  Source=sqlsvil; User ID=%user; Password=%password"
  asklogin="true"/>
```

Sample database connections for Java

Here are some **examples** of database definition in a Java environment:

MSDAccess database

For a Microsoft Access database you can use the Sun JdbcOdbcDriver:

```
<database name="Demo: IpseDixit" schema="" type="MSAccess"
  driver="sun.jdbc.odbc.JdbcOdbcDriver"
  connectionstring="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};
  DBQ=C:\web\public\RemoteQuery\IpseDixit EN.mdb;" />
```

MySQL database

In the following example, the connection is done using the JDBC driver provided by MySQL. The general form of the connection string is the following:

```
jdbc:mysql://[host][, failoverhost...][:port(3306)]/[database][?propertyName1][=propertyValue1][&propertyName2][=propertyValue2]...
```

```
<database name="Demo: Northwind" schema="northwind" type="MySQL"
  driver="com.mysql.jdbc.Driver"
  connectionstring="jdbc:mysql://localhost:3306/northwind?user=RemoteQuery&password=RemoteQuery" />
```

Oracle database

In the following example, the connection is done using the Oracle jdbc driver. The general form of the connection string is the following:

```
jdbc:oracle:thin:[user_id]/[password]@[host]:[port(1541)]:[database]
```

```
<database name="Demo: Scott" schema="SCOTT" type="Oracle"
  driver="oracle.jdbc.driver.OracleDriver"
  connectionstring="jdbc:oracle:thin:REMOTEQUERY/REMOTEQUERY@localhost:1541:TESTDB" />
```

Postgres database

The following example uses the postgresSQL jdbc driver. The connection string has the following form:

```
jdbc:postgresql://[host]:[port(5432)]/[dbname][?propertyName1][=propertyValue1][&propertyName2][=propertyValue2]...
```

```
<database name="RemoteQueryTest" schema="public" type="PostgreSQL"
  driver="org.postgresql.Driver"
  asklogin="true"

  connectionstring="jdbc:postgresql://localhost:5432/RemoteQueryTest?password=%password&
  ;user=%user" />
```

SQLite database^(new)

```
<database name="Demo" schema="" type="SQLite"
  driver="org.sqlite.JDBC"
  connectionstring="jdbc:sq-ite:C:\tmp\demo.sqlite" />
```

SQLServer database

The jdbc driver for SQLServer database has a connection string with the following form:

```
jdbc:sqlserver://[serverName[\instanceName][:portNumber]][;property=value[;property=value]]
```

```
<database name="sqltest" schema="myschema" type="SQLServer"
  asklogin="true"
  driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
  connectionstring="jdbc:sqlserver://localhost:1433;userName=%user;password=%password;
database=remotequery" />
```

Special cases

Specifying a subset of the database tables

The **query** attribute can be specified for restricting the visibility to objects (tables and views) with special characteristics (consider that you may define several “logical” databases pointing to the same “physical” database, provided that you use a different mnemonic name for each).

In the example below, only tables with name beginning with “DW_” are retrieved.

```
<database name="Dept datawarehouse" schema="schema01" type="Oracle"
  connectionstring="Provider=OraOLEDB.Oracle;User ID=xxx;Password=xxx;Data Source=PROD"
  query="SELECT 'T',OWNER,TABLE_NAME FROM ALL_TABLES WHERE TABLE_NAME LIKE 'DW_%' ORDER
BY 2,3"/>
```

That query must always produce a set of rows with three columns:

object type	either T (table) or V (view)
object owner	table view owner (schema name, creator etc.)
object name	table or view name

Specifying a non-supported database

The **query** attribute can be also specified for accessing a database not included in the built-in types (that are Access, SQL Server, Oracle, MySQL and Postgres). To do that (in addition to specifying type="Other" with the proper connection string) you need only to create a SELECT statement returning the same recordset described above. For instance, the query for a DB2 database might be:

```
query ="SELECT TYPE,CREATOR,NAME FROM SYSTEM.SYSTABLES WHERE TYPE IN ('T','V') ORDER BY
1,2,3"
```

Usage of the “asklogin=true” attribute

As shown in the SQLServer example, by specifying **asklogin=true** you make RemoteQuery prompt for userid and password when the iPhone connects to that database. On the contrary, if asklogin is omitted or is equal to **false**, each connection will use the fixed user id and password specified as part of the connection string.

When **asklogin=true**, the connection string **must contain** the proper userid and password parameters (as in the standard case) **with two placeholders** in the place of the fixed user id and password would be specified:

%user will be replaced by the user id entered by the iPhone user

%password will be replaced by password entered by the iPhone user

The two placeholders must be written **lowercase** as shown.

Step 4: Configuring the user authorizations

Users are granted access to databases by means of the definitions in the “**users**” element in the configuration.xml file; here is an example:

```
<!-- List of authorized users -->
<users>
  <user name='rousseau claude' udid='0987676876543417818281882992991999392919' dblist='*' />
  <user name='sommers john' udid='0987676876543417811201020018991999392919'
dblist='demo,db_1,dbtest' />
  <user name='everyone' udid='*' dblist='demo' />
</users>
```

each **user** tag contains the following attributes:

Attribute	Mandatory	Description
name	yes	any mnemonic name for the user (typically his/her real name)
udid	yes	unique device identifier of the iPhone or iPod; it can be obtained using a function in the RemoteQuery application itself (or using iTunes). ---- If udid='*', the entry is applicable to all udid's, that is all RemoteQuery users are allowed to access the databases listed in the dblist attribute.
dblist	yes	Comma separated list of database names the user is allowed to access. Each name must correspond to some database name in the databases element of the same configuration.xml file. ---- If dblist='*', the specified udid is allowed to access all databases included in the databases element.

Step 5: Validating the configuration.xml file

After any change to the configuration.xml file you should verify its format, syntax and content, by using the tool provided.

Validating the configuration file (Windows)

In Windows, simply double click the **validate.vbs** file that is a script provided with the package. It will issue a message saying "Validation OK" or an error message if something is wrong.

Note: the validate.vbs script presumes that the configuration file is in the same folder as the script itself.

Validating the configuration file (Java)

To validate the configuration.xml file on the servlet container, simply call the RemoteQuery servlet with the function VALIDATE:

<http://host:8080/RemoteQuery/RemoteQuery?F=VALIDATE>

Step 6: Configuring the iPhone/iPod clients

Setting the connection(s) to RemoteQuery server(s)

After setting up a server with its server-side components, all iPhone and iPod that wants to access that server need to add the proper connection string to the server.

The connection string is a **http** or **https** URL that is added in the “connection” section of the “Settings”; an example of such string is the one that is preconfigured, for demo purpose, in the installed application:

```
http://rq.logicainformatica.it/demo/RemoteQuery.asp
```

The URL must always point to the .asp page (RemoteQuery.asp) or to the Java servlet (RemoteQuery).

If the site is configured with basic authentication, the host name must be preceded by the userid/password, in the standard syntax (domain\userid:password@), as follows:

```
http://userid:password@rq.logicainformatica.it/demo/RemoteQuery.asp
```

or (with a domain name):

```
http://domain\userid:password@rq.logicainformatica.it/demo/RemoteQuery.asp
```

Getting the device id (UDID)

The device UDID of each iPhone/iPod must be used for proper user configuration, as described in the preceding section; the RemoteQuery application has a special “Device ID” button, in the Settings page for easing the user in sending the UDID via e-mail to a system administrator.

Otherwise, the UDID can be retrieved in iTunes, by clicking on the device identifier label.

Miscellaneous

Tracing (Windows)

The RemoteQuery.asp page is prepared for tracing to file all incoming requests and errors; to enable this feature, the executing user (IIS default user or specific user) must be authorized to write to a Windows folder (by default the same where RemoteQuery.asp is) and a boolean flag (traceOn) must be set in the RemoteQuery.asp (line 91):

```
87: Const SEP = "<br>"
88:
89: ' ATTENTION: enable the following flag only if the current ASP page can be allowed to write onto the
90: '           same folder where it stays. Look at the last function in this file to change the output directory, if needed
91: traceOn = true 'WRITE TRACE TO FILE
92: -----
```

Moreover, if you want to direct the output to another folder, simply change the file path in the TraceToFile function, starting at line 700.

Tracing (Java)

The servlet RemoteQuery can trace to file all incoming requests and errors using the Apache **log4j** tool. The logging is disabled by default; you can activate it and decide the location of log file changing the log4j.properties file located in the **TOMCAT_HOME/webapps/RemoteQuery/WEB-INF/classes** folder.

The first lines of the file contain the settings to activate or deactivate the log:

```
1. #***** Set root logger level to ALL or OFF
2. log4j.rootLogger=OFF, stdout
3. #log4j.rootLogger=ALL, stdout, LOGFILE
4. log4j.appender.LOGFILE.File=/tmp/RemoteQuery.log
```

The line 2. set logger OFF; in the commented line 3., the are the setting to activate the log for all events.

In the line 4. is defined the path of the log file.

See <http://logging.apache.org/log4j/1.2/index.html> for other details.