Techniques for Inferfacing to Databases

A system demonstrating the "How To" can be run from the "Techniques for Interfacing to Databases" section of:

http://www.exsys.com/support/howto

The code for the sample system can be downloaded from the same page.

Database Example

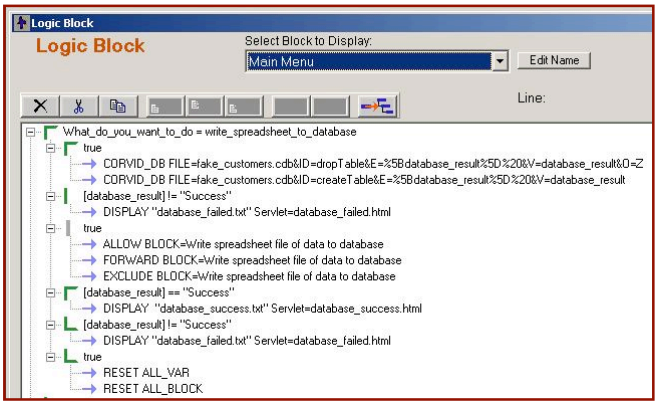
Interfacing to databases can be confusing. This "How To" explains how it works in detail, and includes a demo KB so you can examine the commands and test run it to see how it works.

The demo makes database calls to:

- Get a single variable value
- Get a row of data
- Get a column of data
- Write many rows and columns of data
- Create tables
- Delete tables

Running the Demo

The demo provides a list of database functions that can be run. For the sample, these are simple demonstrations of various types of database functionality commonly used in Corvid systems.



How it Works:

The command block runs the Logic Block named "Main Menu" in forward chaining mode. This means all rules in the Logic Block will execute in order starting at the top and working downward. This Logic Block controls what database action to take based on the user input. The first set of rules will be performed when you select Write the data in a tab delimited text file to the database from the menu.

When you see a CORVID_DB command, you can look for the ID= for the short description of what it does or you can edit the command to see the SQL command that it will perform. This set of rules drops the table if it already exists and creates the table. It writes the data from fake_customers.txt to the table, displays success or a message if there was an error, and finally resets all variables and rules so the system will run again allowing another action.

Looking at it one step at a time. Select the command:

CORVID_DB ...ID=dropTable...

(which is line 3 of Main Menu) and click the “Edit” button, click the “Build Database Command” button, and click on the “Edit” button. You will see the SQL command:

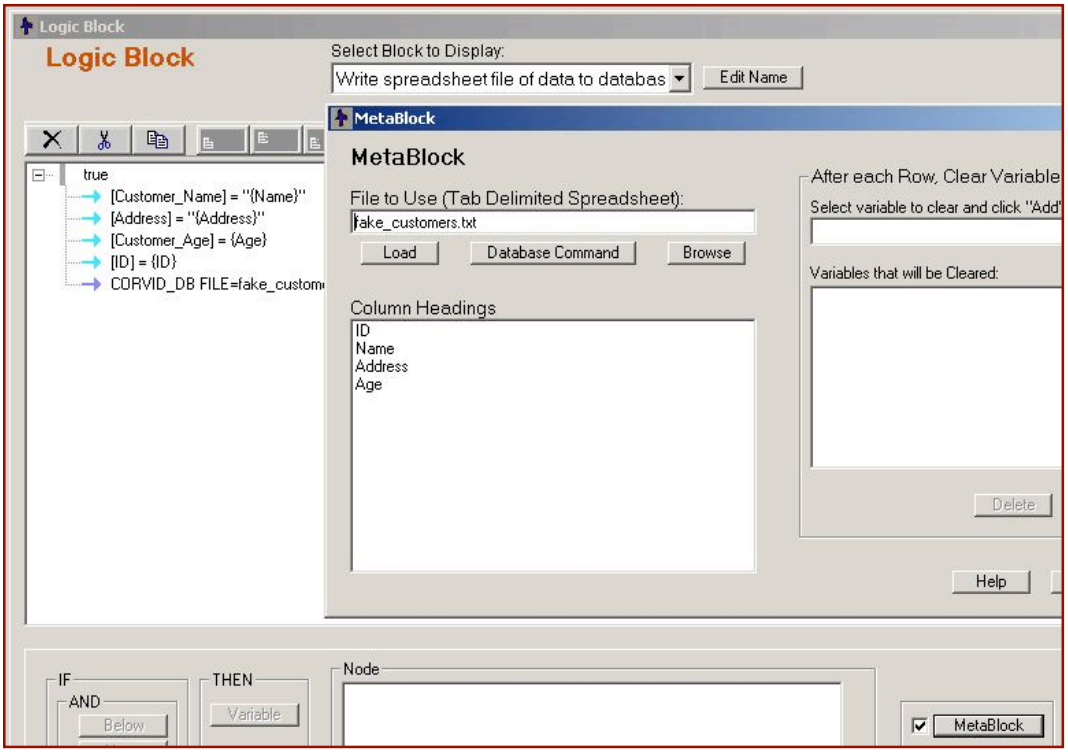
DROP TABLE fake_customers

which deletes the database table named fake_customers, if it exists.

The next database command in “Main Menu” creates that table using:

CREATE TABLE fake_customers...

Both of these database commands are configured to assign a value to the Corvid variable [database_result] if an error occurs. If the commands are executed successfully, the value in [database_result] will be the string “Success”. Lines 5 and 6 in “Main Menu” check to see if the value “Success” was assigned, and if it was not, the error screen “database_failed.txt” will be displayed to provide a customized error message.



Then the Logic Block named “Write spreadsheet file of data to database” is run in forward chaining mode. It uses a Metablock to read each row of data from the file fake_customers.txt (the "spreadsheet") and uses it to add a new row to the database. The Metablock has only one rule, which will be performed on each row of the spreadsheet. The IF condition is “TRUE”, which is a condition that is always true. (“TRUE” will always evaluate to true. Use the expression “TRUE” when the rule should always be performed.) The rule assigns the values from the spreadsheet to variables with similar names. The {Name} is embedded inside quotes to make a syntactically correct formula.

After the embedding, the assignment would look like this, when processing the first row:

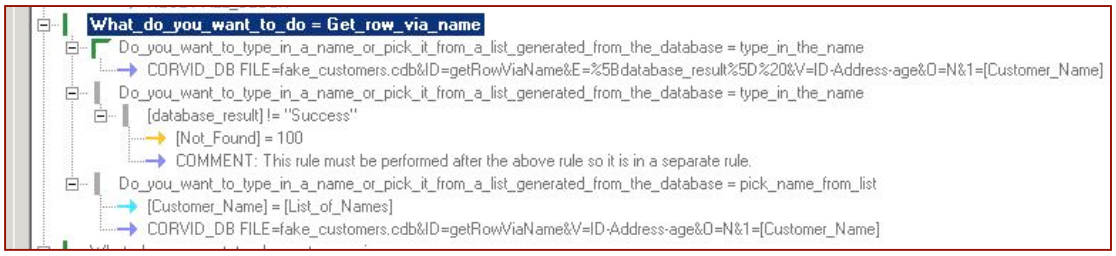
[Customer_Name] = "Ernest"

The database command writes the data to the new row using the SQL command:

INSERT INTO fake_customers VALUES ({N1}, '{E2}', '{E3}', {N4})

where {N1} is replaced by [ID.Format ##], {E2} is replaced with [Customer_Name], and so on. Notice that fields that are text must have ticks around the text. The reason [ID] needs the .Format property is to embed the value as an integer.

If an error occurred during adding the new row, it is displayed by the command in Main Menu on line 14. Line 12 displays a message indicating success. Line 16 and 17 reset all variables and all rules. This allows Corvid to derive or Ask all variables again and fire all rules again.



After the database has been populated with data, you can select “Get the row of data” where the Customer Name matches from the menu question. These rules get data from the row that has the Customer Name that you provide. They ask if you want to type in the name or pick it from a list. If you choose to type in the name, it performs this SQL command:

SELECT ID, Address, Age FROM fake_customers WHERE Name="{E1}"

where {E1} is replaced with the value of [Customer_Name], which is asked. If you choose to pick the name from the list, it uses the same SQL command but [Customer_Name] is first assigned to the value of [List_of_Names], which is a Dynamic List variable that gets its Value List from this SQL command:

SELECT Name FROM fake_customers

Notice it has no WHERE clause so it gets the Name from each row of the database table. Once it has the list of potential values, it asks the user to pick one. After one is selected, that value is assigned to [Customer_Name], which is used in the database command to get the row of data.

Where does Corvid put the row of data? Look in the Database Commands window under the Options tab. Notice the 3 variable names in List Variables to Receive Data. Those names were added to the list using the drop down list box below it and picking the variable(s) and clicking on the Add Below or Add Above buttons.

The last rule gets a single value for a single variable. This is the easiest type of database call. The database call is built into the definition of the variable and is called when Corvid needs to get the value of the variable.

In this demo, the variable is needed because of the:

DERIVE [age]

command but, in most systems, the variable is needed because it was used in a Logic Block or Command Block. In the Variables window, select [age] and notice the database call in Edit Data Source for Value under the Options tab.

The SQL command is:

```
SELECT age FROM fake_customers WHERE name='{E1}'
```

where {E1} is replaced by the value of [List_of_Names]. Since this will cause Corvid to need [List_of_Names], it will get the list of names from the database and ask the user to pick one.

Running the Sample Systems on Your Server

To run the database example:

1. First you must have some database software installed on the server. The database can be any brand that supports ODBC or JDBC connections, such as MySql, Access, Oracle, and almost all other databases.

2. If you will run the KB as a servlet (Corvid Servlet Runtime): Put the example KB files on the server. If you are using Tomcat, put them in a new folder, such as “Database_example” in webapps. You must have CORVID.war installed on the server (in Tomcat or another servlet container such as IBM Websphere), if you are running Corvid Servlet Runtime.

3. If you will run the KB as an applet or application: You must have CorvidDB.war (or the CGI equivalent) installed on the server that contains the database. If you are running as an applet, the server that is running the CorvidDB software must be the same server that serves your KB as an applet. If you are running as an application, the server can be any server. You can install Tomcat onto your computer and then you can run Corvid and CorvidDB in all these different ways. CorvidDB is a middleman program that runs on the server and receives requests from your KB applets or applications for data from the database.

   If running as an applet: Put the example KB files on the server. Where you put the KB files depends on the web page server software. Put them in a sub folder of the "root" folder (which is not the root of the drive).

Server Software	Location of “root” folder
IIS	C:\inetpub\wwwroot
Apache	wwwroot
Tomcat	Webaps/ROOT

   Putting files in the root folder or a sub folder authorizes the server to serve those files.

4. Using your database software, create a database named fake_customers. You do not have to create any tables. For example, if you have MySql, open its command window and type:

   ```
   CREATE DATABASE fake_customers;
   ```

5. If you will use an ODBC connection, create the DSN for the database. It must be a System DSN. Do this in the Control Panel. The exact location depends on the version of Windows being used. On XP it is in:

   Start | Control Panel | Performance and Maintenance | Administrative Tools | Data Sources (ODBC)

6. Open database_example.cvd in Corvid editor. Edit the variable named age. Select the Options tab and click the Database Cmd button. Click the “Edit Connect Data” button. If you are using ODBC, replace your_DSN with the DSN name you created in the previous step. If you are using JDBC, change the Driver and the Connection to what your database software requires.

7. If your database is password protected, enter the User and Password.

8. Click OK to close all the windows until you are back at the Variables window.

9. Copy the .cdb file to the server. If you are using CorvidDB, copy it to the webapps\CorvidDB folder. If you are using Corvid Servlet Runtime, copy it to the KB folder (the folder under webapps that has the .cvR file), if its not already there.

10. Open the .cvd file in Corvid editor and click the Properties button and under the Database tab, change the URL for the database program. If you will be running the Corvid Servlet Runtime, do nothing. If you are running as an applet and using the CorvidDB servlet, hit the Properties button and under the Database tab replace localhost with your host name. Include the port number, if any. If you are running as an applet using the CGI program, replace the entire URL. See the manual for instructions.

11. Save the .cvd file and make sure the files are on the server.

12. You are now ready to run it. Open a browser and go to the URL that runs your KB. The URL will be:

    http://localhost:8080/subfolder/database_example.html

    where you must replace localhost:8080 with your host name and port, replace subfolder with the sub folder used in step 3. Do not include the "root" folder.

    For example, if you put the KB files in webapps/ROOT/Database_example/ on a server running Tomcat, then replace subfolder with Database_example.

    Important: On the first run when the first question is asked pick:

    - Write the data in a tab delimited text file to the database

    This will cause the KB to populate the database with the data that is in the fake_customers.txt file. If it reports an error, verify you did step 6 above. Once the database is full of data, you can select to read various data from the database. The RESULTS screen displays all variables that have a value.