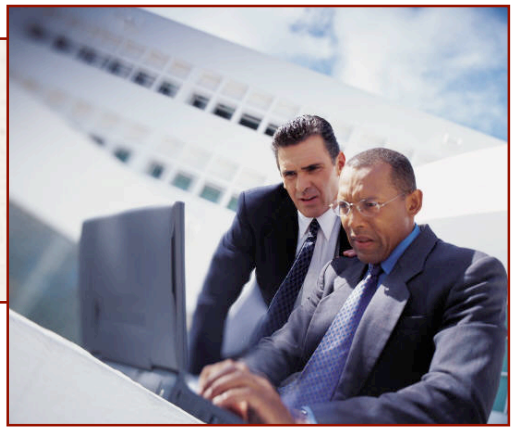
How to Automatically Save User Input to a Database After Every Question

A system demonstrating this "How To" can be run under the "Save User Input to a Database – Applet or Servlet" section of:

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

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



In some systems, it is desirable to allow the end user to answer some questions and quit mid-session, with the ability to return to the same session later. This can be very useful when the system:

- Asks many questions and takes a long time to run
- Asks a question that the end user may not be able to immediately answer
- Requires input from several different users, each providing answers to some questions
- Implements a "model" that can be run in a "What-if" mode, allowing the user to change some of the input values

This approach saves all of the user's input after each question, allowing the user to simply shut down their browser or exit the session without any lost input. (There is also a way to provide the user with a "Save" button that will save data and exit, but that requires them exiting via that button. That approach is covered in a separate "How To" Document.)

Overhead

There is some overhead in saving state, since after each question, the system will send all of the data it has so far, to be stored in the database. This delay and overhead is not noticeable when running using the Corvid Servlet Runtime since it is already running on the server. However, when running using the Applet Runtime, each database call requires calling a server-side program to perform the database operation. This can introduce noticeable delays between questions. If these delays are a problem, it may be better to require the user to save the session by clicking a "Save" button. (See the "How To" document: "How to Save User Input to a Database by Clicking a 'Save' Button".)

User Identifiers

Saving the input requires some way to identify the specific user so that they can return to the session later. This can be provided by a login external to the Corvid system or any other approach that produces a unique ID that they the user can provide when they return to the system. The session data will be stored under that

ID, which is why it must be unique. For many systems the user's email address can be used since it is guaranteed to be unique. If the session data could contain confidential information, or to provide greater security, a user ID/password approach can be used.

Database

Before this technique will work, there must be 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.

Create a database named SaveSession and create a table named SavedSessions. You can do so using these SQL commands:

```
CREATE DATABASE SaveSession;
USE SaveSession;
CREATE TABLE SavedSessions (UserId CHAR(50) NOT NULL, SessionData BLOB
NOT NULL);
```

The table named SavedSessions should have 2 fields of these types:

UserId:	a CHAR() or similar type that is big enough to hold the largest allowed user identification string.
SessionData:	a BLOB or similar type that is big enough to hold all the variable names and their values.

Edit any database command (for example, in the Variables window select [ssa_#SavedSessions] and under its Options tab click the Database Cmd button) and click Edit Connect Data and change the sample values to your actual values.

Change the Driver and Connection strings to whatever your database requires. If your database is Microsoft Access or any other ODBC database, use these:

Driver:	sun.jdbc.odbc.JdbcOdbcDriver
Connection:	jdbc:odbc:your_DSN

where "your_DSN" must be replaced with the System DSN name that you created for that database. The Login and Password are blank unless your database requires them. You can now test run the sample system independently or merge it into your KB.

To Test Run the Sample System

The sample system is designed to be test run independently or can be merged into your existing KB. This example requires Exsys Corvid v3.4.0 or higher.

You need a server that supports servlets. Even if you are running with the Corvid Applet Runtime, it must call a Servlet to perform the actual database commands. These instructions will assume you are using Tomcat on your test computer and the host will be **localhost** on port **8080**.

Create a new folder and put the files from the Sample System folder in it. Open the "SaveSessionAuto.CVD" Corvid file.

Running Your Expert System with the Corvid Applet Runtime

The CorvidDB Servlet must be installed and running to process the database commands. Put the CorvidDB.war file in Tomcat's webapps folder and either deploy it or restart Tomcat. Put the .cdb file in the webapps/CorvidDB/ folder. See the instructions for CorvidDB if you have any problems. In the Properties window under the Test Run tab, uncheckmark Specific URL.

Under the Database tab in the Applet Runtime grouping, specify the URL for the CorvidDB. It could be something like:

http://localhost:8080/CorvidDB/corviddbservlet

Click Done. Click the Run button (the blue triangle).

When you run the system, the first question asks for a User ID. This is used to build a unique identifier for the saved data. You can enter any string, but to avoid reading/overwriting input from someone else, it must be unique. Be sure to remember the ID string entered since you will need it to recover the saved data. Your email address is a good choice for the User ID since it is guaranteed to be unique. (In a production system, there would be a combination of user ID and Password to assure that each user's data is protected. That has not been added to this system to keep it simple.

The next question asks your name. The system will then ask 2 other simple questions. The answers provided do not matter since there is no underlying decision-making logic in the system. For the first time through the system enter a User ID and name, and answer the "Color" question. When the next question is asked, just close your browser window without answering it. Then start the system again and enter the same User ID. The system will tell you that stored data was found for that ID and ask if you wish to recover it. Answer "Yes" and you will immediately be back to the question where you left off. Answer the question, and the results will display both the input from the first session and the additional input in the second session.

Running the Sample System with the Corvid Servlet Runtime

First run the SaveSessionAuto.CVD system with the Corvid Applet Runtime as described above. This is necessary to build the CVR runtime files

Move the system files and templates from the SaveState folder to your server as you would any other Corvid system and run it. (See the Corvid Servlet Runtime manual for details in installing and running the Corvid Servlet Runtime).

Answer the questions the same way as described above for the Applet Runtime.

How it Works

This technique saves all the user's answers for the current run (session) and reads that data back next time the user starts a new run. The data is stored in a database in a table that has two columns (fields). One column is the user's ID and the other column holds all the data of the current run (the SessionData).

The table holds one row (record) for each user running the KB. When the user runs the KB for the first time, a new row is created for that user. From then on, that row will be updated every time the user answers any question in any run of the KB.

The table will have one row for every user that has ever run the KB. The SessionData will be a string (text) that contains all the variables in a format that Corvid can read. It looks similar to this:

```
[name] John Doe
[age] 55
[favorite_color] red
```

Notice that Static List variables will use the short text of the value.

All the SessionData will be a single string that contains Carriage Return characters. The required size (capacity) of the SessionData column is dependent upon the number of variables and the length of their values. To be sure SessionData can hold all the data, declare it to be of type BLOB or MEMO.
When the user runs a system, it calls "RestoreSession", which is a Command Block. It first counts the number of rows that have this user's ID. This is triggered by the command:

```
IF [ssa_#SavedSessions] == 1
```

Since Corvid needs the value of [ssa_#SavedSessions], it performs the database call in External Data Source For Value, which is this SQL call:

```
SELECT COUNT(UserID) FROM savedsessions WHERE UserID='{E1}'
```

where {E1} is replaced with the value of [ssa_User_ID] but with any quotes escaped. It is necessary to escape the quotes of any text inside quotes or ticks. The count can be only 1 or 0. If it is 0, then it creates a new row for this user's ID. If it is 1, then it recovers the SessionData, which means it assigns those variables to the value saved in the database.

It reads the SessionData from the database by using this SQL call:

SELECT SessionData FROM savedsessions WHERE UserID='{E1}'

It creates a new row by using this SQL call:

INSERT INTO savedsessions VALUES ('{E1}', '')

The SessionData is the empty string ''. It will be assigned something after the first question is answered.

As each question is asked, the “After Ask” command is automatically executed. This is the SQL command to write the data up to that point into the database.

To see how this is done, open the Variables window and select the variable [color]. Under the Options tab in the After Ask call, click on the Database Cmd button. Click on the Edit button to see the SQL command:

UPDATE savedsessions SET SessionData='{E1}' WHERE UserID='{E2}'

where {E1} is replaced with the value of [ssa_User_ID] and {E1} is the pseudo variable [~DATA_CR]. The pseudo variable [~DATA_CR] has, as its value, a string that has all the variables that have values and their values in the format similar to this:

[name] John Doe
[age] 55
[favorite_color] red

Merge the Sample System into your KB to Add the Functionality

Note: The sample system can also be merged into your system to provide the core functionality to save/recover input data.

1. Make a copy of the SaveSessionAuto.CVD file and open it.
2. Delete the Logic Block named “Delete this Logic Block” before you Merge.
3. Delete the Command Block named “SampleStartingCmdBlock-Delete” this Command Block before you Merge.
4. Delete the variable [conf].
5. In the Variable window, make sure the “Show Advanced Options” checkbox at the bottom of the window is selected.
6. In the variable window select the variable [color] and click the “Options” tab. At the bottom of the options, there is an “After Ask” command. Copy the text in that edit box and paste it in a Notepad window for future use.
7. Delete the variable [color].
8. Save the modified copy of SaveSessionAuto.CVD and exit Corvid.
9. You are ready to merge the systems. Be sure you make a backup copy of your system’s CVD file. Then open your system CVD file in the Corvid editor. Under the “File” menu select “Merge”. Browse to the copy of the SaveSessionAuto.CVD file that was modified above and select it. It will be merged into your system file.
10. Copy the “After Ask” command text that was pasted into Notepad. This must be pasted in as the “After Ask” command for every question that should automatically save state. For some systems this will be every question. However, due to overhead or question grouping, you may prefer not to save the session after each question is asked, but only after milestone questions. In which case, paste the “After Ask” command only into those variables.
11. Add this command as the first line of your starting Command Block:

    EXEC BLOCK=RestoreSession

To do so, edit your starting Command Block and select the first line and click the Add Above button. Under the Blocks tab check Command Block, pull down its drop down list and select “RestoreSession” and click OK.

12. Save the modified, merged system. It will contain the variables and blocks needed to save sessions.

Note: If the full system is modified in a way that significantly changes the meaning of variables or the questions asked of the user, the saved session data in the database should be deleted, as it may not match the edited system.