Application Note

Manipulating the DBEngine 3.0 Custom Control Interactively in Microsoft's App Studio

Introduction

This Application Note gives some examples of how to use the DBEngine 3.0 Custom Control interactively in Microsoft's App Studio. The intention here is to provide a reference to Microsoft Visual C++ developers on how to use the DBEngine Custom Control, until the official DBEngine 3.0 Custom Control Programmer's Guide - Visual C++ Edition is released.

The examples presented in this Application Note are interactive examples showing how to use the DBEngine Custom Control's properties and database Actions. No C or C++ code is presented here (for example C/C++ code see the DBEngine Custom Control Programmer's Guide - Visual C++ Edition in the DBENG3C.ZIP pre-registration evaluation kit.) The DBEngine 3.0 Custom Control Reference Guide (located in the DBENG3.ZIP file) should be used as a technical reference while performing the examples demonstrated in this Application Note.

Here is a sample outline of the material presented in this Application Note:

I. Interactive Examples

1. Create a database table using the *CreateTable* Action.
2. Create an index for a database table using *CreateIndex*.
3. Open a database table using the *OpenTable* Action.
4. Writing a field using the *PutField* Action.
5. Inserting a record using *InsertRecord*.
6. Moving to the first record in a table using *FirstRecord*.
7. Moving to the next record using the *NextRecord* Action.
8. Moving to the last record in a database using *LastRecord*.
9. Moving to the previous record using *PreviousRecord.*
10. Reading a record using the *GetRecord* Action.
11. Reading a field value using *GetField*.
12. Closing a database table using *CloseTable*.

Examples

Before you begin to interactively manipulate a DBEngine 3.0 Custom Control in the App Studio, you must first create an instance of the DBEngine Custom Control. You can do this by following this procedure:

1. Install the DBEngine Custom Control so that it is available from the App Studio's Control Palette. Select **File** then **Install Controls...** from the App Studio's main menu. Select the DBENG3.VBX file from the appropriate PATH. Then select OK.

2. Create a Dialog Box, then select the DBEngine Custom Control from the App Studio Control Palette and place a DBEngine control on the Dialog Box.

3. Double-click the DBEngine control on the Dialog Box. You will then be presented with **Dialog DBENGINE Properties** window.

4. You will most likely be presented with the **General** property settings. Select the **Styles** property selection from the Combo Box.

5. At this point you should have all of the DBEngine properties described in the DBEngine 3.0 Custom Control Reference Guide in front of you.

The example interactions that follow require you to manipulate the DBEngine properties available in the **Styles** property settings window.

Example 1. CreateTable

This example shows you how to create a Paradox database table using the *CreateTable* Action. To create a database table using the *CreateTable* Action you must first provide information in five specific DBEngine properties:

TableName should have the name of the database table you wish to create including a PATH specifier if needed. Do not place a file extension (.DB) in this property setting.

NFields should be equal to the number of fields present in the database table.

TableFieldNames should have a comma separated list of field names.

TableFieldTypes should have a comma separated list of the field types. The order of the field types is important and must be a one-to-one correspondence with the field names present in the *TableFieldNames* property setting.

TableType should be set to the appropriate table format. Possible settings are shown below:

Value	Description
0	Paradox35
1	Paradox40

Note: See the DBEngine 3.0 Custom Control Reference Guide for more detailed information on the DBEngine properties and Actions.

Once the above property settings have been made you can perform a *CreateTable* Action. Once the *CreateTable* Action is performed, the result (Reaction) of the Action is placed in the *Reaction* property. A zero value is placed in *Reaction* if the Action was successfull, a non-zero (error code) is placed in *Reaction* if something went wrong.

Try this example (in the following examples do not use quotes just type what is inside the quotation marks in the App Studio's property value text field):

1. For the *TableName* property type in "C:\TEST"
2. For the *NFields* property type in "4"
3. For the *TableFieldNames* property type in "ID,Last Name,First Name, MI"
4. For the *TableFieldTypes* property type in "N,A30,A20,A1"
5. For the *TableType* property select "0-Paradox35"
6. For the *Action* property select "8 - CreateTable"
7. If the table was created successfully the *Reaction* property setting will equal 0.

This example if successfull, created a Paradox database table called TEST on drive C, in the root directory. The TEST database table created has the following structure:

Field	**Field Type**
ID	N
Last Name	A30
First Name	A20
MI	A1

The TEST database table consists of four (*NFields* = 4) fields. We will continue to use this TEST database in our following examples.

Example 2. CreateIndex

The *CreateIndex* Action is performed to create indexes for database tables. When using the DBEngine 3.0 Custom Control there are two types of indexes that can be created, Primary and Secondary indexes (for detailed information concerning indexes see the DBEngine 3.0 Custom Control Reference Guide.)

In this example we will create a Primary index that consists of a single key field (ID). Prior to performing the *CreateIndex* Action several other DBEngine properties must be set with valid information:

TableName must have the name of the database table that the index is being created for.

IndexNFields must have the number of key fields composing the index.

IndexID must have the field number of the key field. For Primary indexes *IndexID* is always 1. For secondary indexes, it is the field number of the key field (for detailed information concerning indexing and key fields see the DBEngine 3.0 Custom Control Reference Guide.)

IndexType should be set to:

0 - Primary
1 - NonMaintainedSecondary
2 - MaintainedSecondary

Try this example (in the following examples do not use quotes just type what is inside the quotation marks):

1. Set the DBEngine control's *TableName* to "C:\TEST" if it is not already so set.
2. Set the *IndexNFields* property to "1"
3. Set the *IndexID* property to "1"
4. Set the *IndexType* property to "0 - Primary" (use the property value combo box for predefined selections.)

5. For the *Action* property select "7 - CreateIndex"
6. If the index was created successfully the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

We set *IndexID* equal to one because we wish to create an index with one and only one key field. We set *IndexID* to one because our key field is the ID field which is the first field in the database record structure. We set *IndexType* to 0 - Primary because we wanted to create a Primary index.

The TEST database table now has the following structure:

Field	**Field Type**
ID	N*
Last Name	A30
First Name	A20

MI A1

Note: where key fields are marked by an *.

Example 3. OpenTable

The *OpenTable* Action is used to open a database table for processing. The *OpenTable* Action expects the following properties to be set with valid informaton:

TableName has the name of the database table to open.

IndexID has the field number of the index to be opened along with the table (use zero (0) to open the table with all associated indexes.

SaveEveryChange should be set to one of the following:
0 - False (buffers data to disk)
1 - True (save changes to disk immediately)

. This example will open our C:\TEST database. We will buffer changes to disk, and open the table with all associated indexes.

Try this example (in this example do not use quotes just type what is inside the quotation marks):

1. Set the DBEngine control's *TableName* to "C:\TEST" if it is not already so set.
2. Set the *IndexID* property to "1"
3. Set the *SaveEveryChange* property to "0 - False"
3. For the *Action* property select "37 - OpenTable"
4. If the index was created successfully the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

When a database table is successfully opened, the default current record is the first record in the table image. non the less we will next demonstrate how to move to the first record in the database table regardless of where the current record pointer is located.

Example 4. PutField

In order to write values into a database table using the DBEngine 3.0 Custom Control, you need to first place values into the database fields, then either insert a record into the database table or update the current record in the database

table (you can also append a record). Storing data or writing data to the table is thus a two-step process:

1. Place the information into the field(s) of the DBEngine's record buffer.
2. Insert, append, or update the record in the database.

The *PutField* Action requires a valid open database table. It also requires valid information in the following DBEngine properties:

FieldName holds the name of the field you are working with.

FieldValue holds the value you intend to write into the field.

This example will show you how to place values into the fields of the open C:\TEST database table. The next example will show how to write the record to the table.

Try this example (in this example do not use quotes just type what is inside the quotation marks):

Note: The C:\TEST table must be open!

1. Set the *FieldName* property to "ID"
2. Set the *FieldValue* property to "11223"
3. Set the *Action* property to "40 - PutField" (use the property value combo box)
(If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.)
// We just placed info into the ID field, let's do the Last Name field Next.

4. Set the *FieldName* property to "Last Name"
5. Set the *FieldValue* property to "Bebber"
6. Set the *Action* property to "40 - PutField" (use the property value combo box)
(If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.)
// We just placed info into the Last Name field, let's do the First Name field Next.

7. Set the *FieldName* property to "First Name"
8. Set the *FieldValue* property to "Douglas"
9. Set the *Action* property to "40 - PutField" (use the property value combo box)
(If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.)
// We just placed info into the First Name field, let's do the MI field Next.

10. Set the *FieldName* property to "MI"
11. Set the *FieldValue* property to "A"
12. Set the *Action* property to "40 - PutField" (use the property value combo box)

(If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.)
// We just placed info into the MI field, all the fields now have values.

If everything went well, we encountered no errors and are now ready to insert the record into the database table by performing an *InsertRecord* Action.

Example 5. InsertRecord

One values have been written to fields by performing *PutField* Action(s), a record can be inserted into a database table by performing an *InsertRecord* Action. To be successfull, the *InsertRecord* Action requires that the DBEngine control be associated with a valid, open database table. The field values must have been placed into the DBEngine control by performing *PutField* Action(s).

This example will insert the record set up in the previous (PutField example) into the open C:\TEST database table:

1. Set the *Action* property to "25 - InsertRecord" (use the property value combo box).
2. If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

Note: Perform the above two examples again, this time write different field values to the table (for example: ID = 12984, Last Name = "Doe", First Name = "John", and MI = "D".) This will give us multiple records in the C:\TEST database. We will need more than 1 record when we start to perform the DBEngine Actions that move from record-to-record.

Example 6. FirstRecord

The *FirstRecord* Action can only be performed after a table has been successfully opened via the *OpenTable* Action. If the table has been successfully opened, you can move to the first record in the table image at any time by performing a *FirstRecord* Action. The *FirstRecord* Action has no other requirements or prerequisites other than it be performed on an open table.

The following example shows how to perform a *FirstRecord* Action:

1. The DBEngine control must have a valid open table (if you just opened the C:\TEST table by doing the previous example you have a valid open table. If not, do the above example(s) in order to successfully open the C:\TEST database

table.)

2. For the *Action* property select "15 - FirstRecord"
3. If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

Example 7. NextRecord

The *NextRecord* Action can only be performed after a table has been successfully opened via the *OpenTable* Action. If the table has been successfully opened, you can move to the next record in the table image at any time by performing a *NextRecord* Action. The *NextRecord* Action has no other requirements or prerequisites other than it be performed on an open table.

The following example shows how to perform a *NextRecord* Action:

1. The DBEngine control must have a valid open table (if you just opened the C:\TEST table by doing the previous example you have a valid open table. If not, do the above example(s) in order to successfully open the C:\TEST database table.)

2. For the *Action* property select "33 - NextRecord"
3. If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

Example 8. LastRecord

The *LastRecord* Action can only be performed after a table has been successfully opened via the *OpenTable* Action. If the table has been successfully opened, you can move to the last record in the table image at any time by performing a *LastRecord* Action. The *LastRecord* Action has no other requirements or prerequisites other than it be performed on an open table.

The following example shows how to perform a *LastRecord* Action:

1. The DBEngine control must have a valid open table (if you just opened the C:\TEST table by doing the previous example you have a valid open table. If not, do the above example(s) in order to successfully open the C:\TEST database table.)

2. For the *Action* property select "29 - LastRecord"

3. If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

Example 9. PreviousRecord

The *PreviousRecord* Action can only be performed after a table has been successfully opened via the *OpenTable* Action. If the table has been successfully opened, you can move to the previous record in the table image at any time by performing a *PreviousRecord* Action. The *PreviousRecord* Action has no other requirements or prerequisites other than it be performed on an open table.

The following example shows how to perform a *PreviousRecord* Action:

1. The DBEngine control must have a valid open table (if you just opened the C:\TEST table by doing the previous example you have a valid open table. If not, do the above example(s) in order to successfully open the C:\TEST database table.)

2. For the *Action* property select "38 - PreviousRecord"
3. If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

Example 10. GetRecord

Before you can read individual fields of information from a database table you need to perform a *GetRecord* Action. Reading data from a table is a two-step process:

1. Get the current record from the database table.
2. Read individual fields of information from the record read in in step 1.

The *GetRecord* Action requires that there be a valid open database table associated with the DBEngine control. This example shows how to perform a *GetRecord* Action:

1. The DBEngine control must have a valid open table (if you just opened the C:\TEST table by doing the previous example you have a valid open table. If not, do the above example(s) in order to successfully open the C:\TEST database table.)

2. For the *Action* property select "21 - GetRecord"
3. If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

Now that we have read-in the current record from the database table, we can move on to the next example which reads in a field from the current record.

Example 11. GetField

Once a record has been read-in to the DBEngine control, you can examine individual field values by performing *GetField* Action(s). You can not expect to obtain valid field values unless you have first performed a *GetRecord* Action (this is a common mistake that beginning DBEngine programmers make, they try to read a field value without first obtaining the record via *GetRecord*.)

Obviously, the *GetField* Action requires a valid open database table. The following property must be set with valid information before performing a *GetField* Action:

FieldName holds the name of the field you wish to examine.

The following steps show how to successfully perform a *GetField* Action:

1. The DBEngine control must have a valid open table (if you just opened the C:\TEST table by doing the previous example you have a valid open table. If not, do the above example(s) in order to successfully open the C:\TEST database table.) Also you must have previously performed a *GetRecord* Action.

2. Set the *FieldName* property to "ID"
3. For the *Action* property select "17 - GetField"
4. If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.

Now for our last example in this Application Note, we will show how to close a database table. You should always close a table when you are through processing it by performing a *CloseTable* Action.

Example 12. CloseTable

This last example shows how to close a database table associated with a DBEngine control. Obviously, before we can successfully close the table it must

be a valid open table. This is the only prerequisite to performing a *CloseTable* Action (the DBEngine control must be associated with a valid open database table.)

The following steps show how to close an open table using the *CloseTable* Action:

1. The DBEngine control must have a valid open table (if you just opened the C:\TEST table by doing the previous example you have a valid open table. If not, do the above example(s) in order to successfully open the C:\TEST database table.)

2. For the *Action* property select "5 - CloseTable"
3. If successfull the *Reaction* property setting will equal 0. If not the appropriate error code will be present in the *Reaction* property.
.

This concludes the Application Note. For more detailed technical information concerning the DBEngine 3.0 Custom Control consult the **DBEngine 3.0 Custom Control Reference Guide**. For extensive information concerning DBEngine programming including example source code see the **DBEngine 3.0 Custom Control Programmer's Guide - Visual C++ Edition**.

End of Document