Writing a Query Definition Function

When the **Data Access Manager** creates a **QueryRecord**, it calls the query definition function specified by the *queryProc* field in the **QueryRecord**. The purpose of the query definition function is to modify the query and the **QueryRecord** before the query is sent to the data server. The query definition function can use dialog boxes to request information from the user. Because a query document is most useful if it can be used by many different applications, no query definition function should depend on the presence of a particular application.

If you want to include a query definition function, you must make it the first piece of code in a resource of type 'qdef' in the query document.

Here is a function declaration for a query definition function.

```
pascal OSErr  MyQDef (long sessID, QueryHandle query);
```

If the application has already initiated a session with the data server, the **DBStartQuery** function passes the session ID for that session in the *sessID* parameter to the query definition function. If the query definition function receives a 0 in this parameter, then the **Data Access Manager** has not initiated a session. In this case, the query definition function can return a 0 in the *sessID* parameter, or it can call the **DBInit** function to initiate a session and then return the session ID in this parameter.

If the query definition function returns a 0 in the *sessID* parameter, the **DBStartQuery** function calls the **DBInit** function and then calls the **DBSend** function to send a query to the data server. If the query definition function returns a session ID in this parameter, the **DBStartQuery** function calls the **DBSend** function immediately.

The query parameter to the query definition function specifies a handle to the **QueryRecord**. The query definition function can modify any of the fields in the **QueryRecord**, including the *currQuery* field that specifies which query is to be sent to the data server. In addition, the query definition function can modify an existing query or create a new query, adding the handle to the new query to the query list. Note that, because a query in memory consists only of a 2-byte length value followed by a character string, the query definition function has to know the exact contents and structure of a query in order to modify it.

The query definition function must return the noErr result code as the function result if the function executed successfully. If it returns any other value, the **DBStartQuery** function does not call the **DBSend** function. The query definition function can return any result code, including noErr, userCanceledErr, or rcDBError.

When the **DBStartQuery** function calls the query definition function, the current resource file is the file that contains the 'qrsc' resource from which the **Data Access Manager** created the **QueryRecord**. When the query definition function returns control to the **Data Access Manager**, the current resource file must be unchanged.

The query definition function can allocate memory and use the dataHandle field

in the**QueryRecord** to store a handle to it. The query definition function must free any memory it allocates before terminating.

The Listing below shows a query definition function that uses a dialog box to prompt the user for a user name and password and then modifies the **QueryRecord** accordingly.

```
// A query definition function
// Assuming inclusion of <MacHeaders>

#include <DatabaseAccess.h>
#include <string.h>

#define    myNameItem                  7
#define    myPassWordItem              8

pascal OSErr MyQDef (long *sessID, QueryHandle query);
pascal Boolean myNamePasswdFltrFunc (DialogPtr theDlg, EventRecord
*theEvent, short *itemHit);

pascal OSErr MyQDef (long *sessID, QueryHandle query)
{
    short               myNumRes;
    ResListHandle myResList;
    ResListPtr     myResLPtr;
    short               myIndex;
    DialogPtr          myDialog;
    short               myDlogID;
    short               itemType;
    Handle          itemHName;
    Handle          itemHPasswd;
    Rect                itemBox;
    Str255              mySTR[2];
    short               itemHit;
    OSErr               myQErr;


    // If sessID = 0, no session has been
    // initiated.  Your qdef may optionally initiate a
    // session, or it can let the DBStartQuery routine take
    // care of this.  In this example, the qdef doesn't
    // check the sessID parameter.

    HLock((Handle) query);
    myNumRes = (*query)->numRes;
    myResList = (*query)->resList;
    HLock((Handle) myResList);
    myResLPtr = *myResList;
    myIndex = 0;

    // Look for a 'DLOG' resource
    while ( (myIndex < myNumRes) || (myResLPtr[myIndex].theType !=
'DLOG') )
        myIndex = myIndex + 1;
```

```
// Was a 'DLOG' resource found, or did the index run out?
if (myIndex < myNumRes)
    myDlogID = myResLPtr[myIndex].id;
    // We found the 'DLOG' resource.
else {
    // The 'DLOG' wasn't found; exit with no error.  This
    // is probably okay; it just means that the query
    // and the query record don't get modified.
    return noErr;
    HUnlock((Handle) query);
    HUnlock((Handle) myResList);
    return;
}

// Found the 'DLOG' and its ID; now put up the dialog box.
myDialog = GetNewDialog(myDlogID, nil, (WindowPtr) -1);
SetPort((GrafPtr) myDialog);

// Now you can change the query record or the query itself.
// What you change is entirely up to you. In this example,
// the qdef changes only the user and password fields
// of the QueryRecord.

GetDItem(myDialog, myNameItem, &itemType, &itemHName, &itemBox);
GetIText(itemHName, mySTR[1]);
GetDItem(myDialog, myPassWordItem, &itemType, &itemHPasswd,
        &itemBox);
GetIText(itemHPasswd, mySTR[2]);

// Make available to the filter routine the strings
// we want the user to edit.
((WindowPeek) myDialog)->refCon = (long) mySTR;

// myNamePasswdFltrFunc is a routine that allows the user to edit
// the name and password fields in the dialog box

ModalDialog(myNamePasswdFltrFunc, &itemHit);
if (itemHit == ok) {

    // The user clicked the OK button. Update the user and password
    // fields of the QueryRecord.

    strcpy ((char *) (*query)->user, PtoCstr (mySTR[1]));
    CtoPstr ((char *) (*query)->user);

    strcpy ((char *) (*query)->password, PtoCstr (mySTR[2]));
    CtoPstr ((char *) (*query)->password);
    return noErr;
}
else
    return userCanceledErr;
HUnlock((Handle) query);
HUnlock((Handle) myResList);
DisposDialog(myDialog);
```

```
}
```