Using the Data Access Manager

There are at least three different ways in which you can use the **Data Access Manager** to communicate with a data source.

- You can use low-level interface routines to send queries and retrieve data from the data source. In this case, your application must be capable of preparing a query in a language appropriate for the data server.

- You can use high-level interface routines to send queries and retrieve data from the data source. In this case, you must have one or more query documents provided by another application.

- You can create your own query documents and use high-level interface routines to send queries and retrieve data from the data source. In this case, your application must be capable of preparing a query, but it can use the same query repeatedly once it has been prepared.

This section describes how to call **Data Access Manager** functions asynchronously. **Using the High-Level Interface** and **Using the Low-Level Interface** describe using each of the interfaces to the **Data Access Manager** to send queries to a data server. ; **Writing a Status Routine for High-Level Functions** describes how to determine the status of the high-level functions at various points in their execution (and cancel execution if you so desire); **Getting Information about Sessions in Progress** explains how to obtain information about **Data Access Manager** sessions that are in progress; and **Processing Query Results** describes how to retrieve query results and convert them to text.

Executing Routines Asychronously

All of the **Data Access Manager** low-level routines and some of the high-level routines can execute asynchronously-that is, the routine returns control to your application before the routine has completed execution. Your application must call the **WaitNextEvent** function periodically to allow an asynchronous routine to complete execution.

> **Note:** The database extension is responsible for implementing asynchronous execution of **Data Access Manager** routines. For example, if you call the **DBSend** function to send a query to a data server, and the database extension calls a device driver, the database extension can return control to your application as soon as the device driver has placed its routine in the driver I/O queue. If you attempt to execute a routine asynchronously and the database extension that the user has selected does not support asynchronous execution, the routine returns a result code of rcDBAsyncNotSupp and terminates execution.

All **Data Access Manager** routines that can execute asynchronously take as a parameter a pointer to a parameter block known as the *asynchronous parameter block.* If this pointer is NIL, the function is executed synchronously-that is, the routine does not return control to your application

until execution is complete.