Chapter 1 — *Protocol API Overview*

Protocol API classes

The following API classes are used to develop a Protocol API application that can interface with the db-One server and search engine.

- **MPNewRequest class** — the class that performs server operations.
- **MPResponse class** — the class that gets the results of server operations.
- **MPSocket class** — the class that encapsulates the socket.
- **MPAuthen class** — the class that authenticates the User ID and password.

Using the Protocol API classes

All of the functionality of db-One exists in the server operations performed by the db-One server. You'll use the Protocol API Request and Response classes only to request that a server operation be performed and to receive results from the server. Both the Request and Response classes define Send(MPSocket s) and Receive(MPSocket s) member functions that you'll use as the mechanism for communication between the client and the server.

To perform any of the server operations you'll need to follow the same basic steps.

- *First*, set the parameters of the request object for the server operation you want performed. The server operation name should be stored in the public variable mp_op.
- *Then* call the Send function of the request object to transmit the request to the server.
- *Last*, call the Receive function of the response object to receive the response from the server.

The Request and Response classes are used only to send the server requests and receive its responses. They have no other purpose.

Note: Server operations are shown in all capital letters in this guide. (For example, MP_BEGIN_SEARCH.)

Additional functions needed

Before using the request and response classes, you must establish a connection to the server and initialize Windows® sockets. To do this, you need to use the following functions:

- **ConnectTCP** — connects to the server and creates a socket.
- **WSAStartup** (a Windows Socket function) — initiates the use of the Windows Socket DLL by a process. Its format is:

```
int WSAStartup (WORD wVersionRequested,
                LPWSADATA lpWSAData);
```

Before exiting your application, you'll need to use the following function:

- **WSACleanup** (a Windows Socket function) — terminates use of the Windows Socket DLL. Its format is:

```
int WSACleanup();
```

For additional information about WSAStartup and WSACleanup, see your Microsoft® Windows help file.

What is a search context?

In order to fully understand the Protocol API, you'll need to understand the concept of a search context and how it is used in db-One.

The search context is the way that search sessions are identified and tracked by the server. It enables the server to distinguish between search requests from different clients. A search context is maintained on the server for each search request from each client.

When you create a search context, your client application receives a unique identifier. Then, any calls your program makes after the search context is established reference the search context you've created. Each client can set up multiple search contexts as long as the client application maintains the search context Id (the search handle).

Search context states

A search context can have two different states: an initial state and a post-search state. The *initial* state exists immediately after the search context is created. At that point, the only information the search context contains is the

search context Id (search handle). The *post-search state* exists after a successful search has been done. At that point, the search context contains a search result list as well as the search context Id. You can then use various server operations (such as MP_GET_FIRST and MP_GET_NEXT) to let the client retrieve manageable sections of the result list.

Server operations that affect the search context

You must call the server operation MP_STOP_SEARCH to end the search session and destroy its search context. Then, call the MP_QUIT server operation, which disconnects from the server. If you don't destroy the search context, memory will be consumed needlessly, potentially affecting other clients who are trying to establish search sessions.

Server operation categories

Server operations are divided into four categories:

- **Connection/search context handling operations** — operations that establish or terminate connection to the server, and operations that set up and manage the search context.
- **Operations for posting search criteria** — operations that post search criteria and perform the search.
- **Search context operations** — operations, such as MP_GET_NEXT (get next matching set of records) that work only following a search.
- **Operations that are independent of any search context** — operations that are not dependent on having performed a search. They are stateless; therefore, they can be done at any time before, during, or after a search.

These operations are described in detail in Chapter 2, "Server Operations" in this guide.

The flow of server operations

This section describes the order in which server operations must be done to perform a basic search. It provides a flow for the most common server operations initiated by Protocol API calls.

A basic search flow

To perform a simple search, you'll need to request the following server operations, in the order indicated:

MP_BEGIN_SEARCH (creates a search context)

MP_START_SEARCH	(performs a search)
MP_GET_FIRST, MP_GET_LAST, or MP_GET_ALL	(gets first, last, or all sets of matching records. You must execute one of these operations.)
<Other operations from the search context category>	
MP_STOP_SEARCH	(destroys the search context)
MP_QUIT	(disconnects from the server)

Important: If you execute an MP_SORT (a search context operation), you'll need to execute either an MP_GET_FIRST, MP_GET_LAST, or MP_GET_ALL operation immediately after the sort. You can then call any of the other search context operations.

Executing the search context operations MP_GET_NEXT or MP_GET_PREVIOUS without first executing an MP_GET_FIRST, MP_GET_LAST, or MP_GET_ALL operation can lead to unpredictable results.