ibi™ WebFOCUS® Reporting Server

Extender for z/OS Db2

Version 9.0.0 and later | April 2024

Contents

Introducing the ibi WebFOCUS Extender for Db2

This section provides an overview of the ibi™ WebFOCUS® Extender for Db2. It also describes the features and summarizes the functions of the Extender for Db2.

In the remainder of this manual, the WebFOCUS® Extender for Db2 will be referred to as the Extender for Db2 or, simply, the Extender.

What Is the ibi WebFOCUS Extender for Db2?

The Extender for Db2 is a member of the Server family of products. The Extender for Db2 provides client Db2 applications on z/OS platforms (such as QMF, DXT, AS, DIS, or Language Access applications) with transparent access to remote data sources through servers. These data sources can be either relational tables and views or non-relational files from any of the data adapters supported by the server. The Extender for Db2 provides this access through Application Services and Network Services.

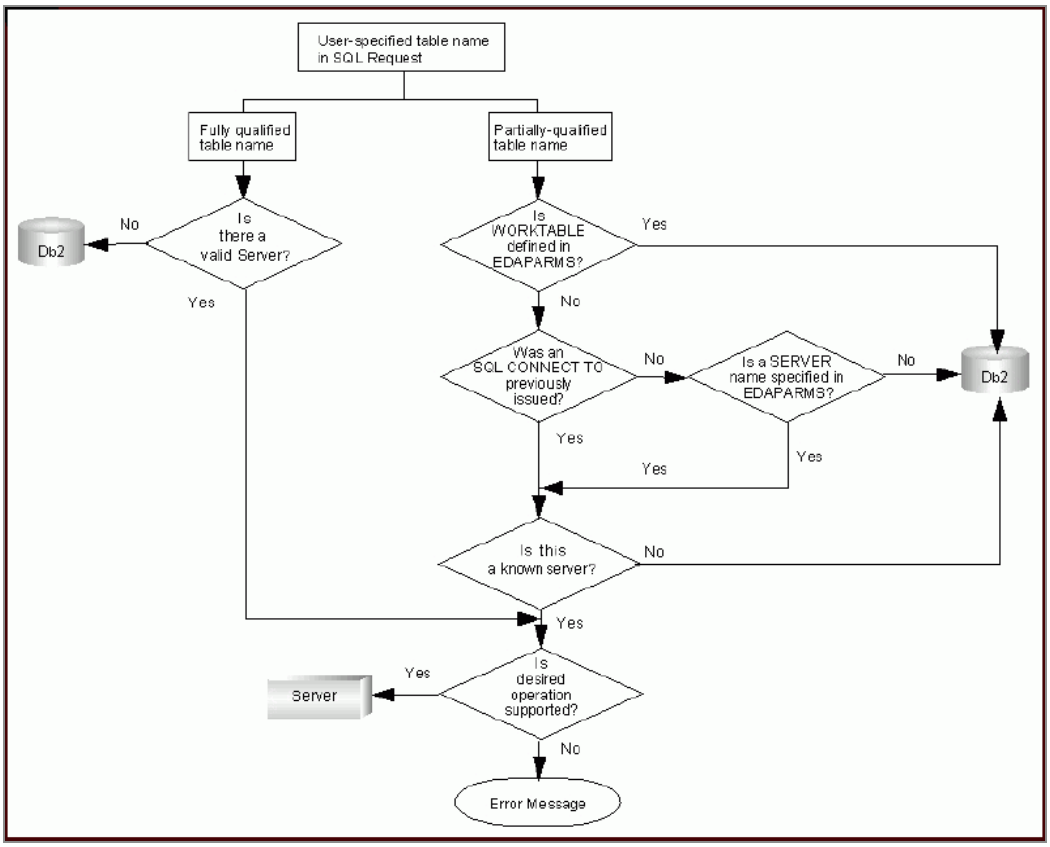
- Application Services provides a standard application programming interface through which client applications, including Db2 applications, send requests to servers and receive data in return. There is no need for client applications to be programmed to the details of the underlying operating systems, networks, or data sources of the server network.
- Network Services provides communications between connected client/server environments. It shields client applications from details associated with sending or receiving information across proprietary communications networks. Network Services on the client comprise the Communications Subsystem/3 (CS/3), supporting the TCP/IP protocol.

Extender for Db2 Environment

The following components make up a server environment with a Db2 application querying data sources:

- A *Db2 application* is an application that accesses data in a local Db2 subsystem.
- The *Extender for Db2* provides access to a server. It takes incoming SQL statements from a Db2 application and either passes them to the local Db2 subsystem, or converts them into appropriate server commands.
- The *Application Service* receives the API calls and passes the request to the server in an appropriate form.
- *Network Services* provides the communications interface for transporting requests to the server and for receiving returned data.
- The *Server* processes the request and passes the resulting information back to the client.

This figure illustrates how these components fit into a Db2-Server environment.



The Db2-Server environment comprises the user application (Db2 application products, such as QMF and DXT, 3GL programs, or SQL tools), the Extender for Db2, and the server. It runs in its own address space. The server also runs in its own separate address space.

As shown in the figure above, the Db2 application can query data residing in the same Db2 subsystem in one of two ways:

- Using the Extender for Db2 and the server.
- Using the local Db2 subsystem through the Extender for Db2, bypassing the server.

The path on which the originating SQL request is directed (local Db2 or server) depends on the proper parsing of the table names.

Features of the ibi WebFOCUS Extender for Db2

The Extender for Db2 provides:

- Support of all SQL functions related to static or dynamic SQL directed at the local Db2 subsystem.

  > **Note:** *Static* SQL calls require precompiling, binding from a Database Request Module (DBRM) to a static plan, and running a load module through a static plan. *Dynamic* SQL calls are constructed and prepared into an executable object that contains access paths to Db2 databases. With dynamic SQL calls, the SQL request string is available to the Extender for Db2.

- Support of dynamic SQL functions for SELECT statements directed at the server. The dynamic SQL functions supported for SELECT statements are: PREPARE, DESCRIBE, OPEN, FETCH, and CLOSE.
- Support of dynamic SQL functions for non-SELECT statements directed at the server. The dynamic SQL functions supported for non-SELECT statements are: PREPARE, EXECUTE, and EXECUTE IMMEDIATE. The list of supported non-SELECT statements is limited to the functionality (Release level) of the server being utilized.
- Support of general-purpose SQL statements (for example, COMMIT, ROLLBACK, and CONNECT).
- Operation using either the IBM Call-Attach Facility (CAF) or the IBM DSN Command Processor as application or call-level interfaces.
- Transparent interface.

Additional features include:

- **EDAPARMS.** This parameter file enables you to set the default error SQLCODE, to

specify a default server for unresolved partially qualified tables, to define worktables, and to enable only standard Db2 operation.

- **Explicit or implicit SQL CONNECT verbs.** Enables the application to explicitly or implicitly CONNECT to a server or a local Db2 subsystem.

Functional Overview of the ibi WebFOCUS Extender for Db2

The Extender for Db2 operates as follows. It:

1. Step 1. Intercepting Dynamic SQL Calls
2. Step 2. Determining Table Location
3. Step 3. Constructing and Dispatching Requests
4. Step 4. Returning Data to the Application

Step 1. Intercepting Dynamic SQL Calls

Db2 applications communicate with their local Db2 subsystems using calls to interface modules. The Db2 application invokes a Db2 entry point pertinent to the communications mode used by the application. The table below describes the standard Db2 database entry points and their corresponding mode of database communication:

Database Entry Point	Mode of Database Communication
DSNALI	Db2 Call Attach Facility
DSNELI	Db2 DSN Command Processor (Db2 TSO Attach Facility)
DSNTIAR	Db2 Error Message Formatting Facility

The Extender provides database entry points with names equivalent to the above standard names. Since the Extender resides between the application and the local Db2 subsystem, database calls made to the local database are intercepted by the Extender entry points.

The Extender for Db2 provides a Db2-like interface to the applications. Depending on the application's standard mode of communication with the local Db2 subsystem, it invokes the Extender entry point instead of the standard database entry point. Four common modes of Db2 entry are supported:

- **Call Attach Facility (CAF).** The application program invokes an Extender for Db2 DSNALI (DSNHLI2) replacement. In this mode, the application program can load this entry point dynamically in its own address space before the entry point's invocation.
- **DSN Command Processor (TSO Attach Facility).** The application program invokes an Extender for Db2 DSNELI (DSNHLI) replacement. With TSO Attach, this entry point is usually statically linked to your application.
- **Db2 Error Message Formatting Facility.** Db2 applications use this component to obtain message text information from Db2 SQLCA return codes and tokens. The Extender provides a DSNTIAR replacement, which is either dynamically loaded or statically linked, depending on how your application uses the standard Db2 DSNTIAR entry point.

Consult the standard database installation of your application to verify which mode of database communication your application requires.

The Extender then determines where to dispatch the database call (to a server or the local Db2 subsystem) to locate the data source. This data source determination is described in the next section.

Step 2. Determining Table Location

The Extender for Db2 uses table names to determine the location of the table in an SQL request. Within an SQL request, a user can specify access to a server data source or a local Db2 subsystem table by using table naming conventions. Table names are categorized as either fully qualified (three-part) names, or partially qualified (two-part or one-part) names. For more information on table naming conventions, see Using the ibi WebFOCUS Extender for Db2 and SQL.

When the Extender for Db2 parses the user's SQL request, it determines whether to dispatch the request to the local Db2 subsystem or to accessible databases residing under a server. This determination is transparent to the user.

After the destination is determined for each table, the Extender for Db2 determines if every table in the SQL statement is consistently destined for either the local Db2 subsystem or a server:

- If every table is consistently destined for a local Db2 subsystem, then the request is simply passed to the Db2 DSNALI module.
- If every table is consistently destined for a server, then the appropriate commands are constructed and a communications dialog is established to dispatch the request to a server. (See Step 3. Constructing and Dispatching Requests for more information about constructing local requests.)
- If the SQL statement references tables that are destined for both the local Db2 subsystem and a server, that request is rejected and the Extender for Db2 generates an error message.

  The Extender for Db2 does not support SQL SELECT statements that contain tables from mixed destinations (such as a local subsystem table and a server table). This type of SQL request is rejected and an error message is generated.

  To join tables from the local Db2 or SQL subsystem with any server-accessible database tables, define appropriate Master Files and Access Files for those Db2 tables under a common Hub Server.

Determining Object Destination Logic

For each SQL request, the Extender for Db2 first determines the destination for each table (either the local Db2 subsystem or a server). To do this, the Extender for Db2 uses the decision tree summarized here and illustrated graphically in the following flowchart.

1. Is the referenced table a:
   - Fully qualified table name?

     Then go to Step 2.
   - Partially qualified table name?

     Then go to Step 3.
2. Does the three-part name reference a:
   - Valid server?

     Then go to Step 5.
   - Undefined server?

     Then send the request to the local Db2 subsystem.
3. Is the partially qualified table name:

- Defined in EDAPARMS under the WORKTABLE keyword?

  Then send the request to the local Db2 subsystem.

- Not defined in EDAPARMS?

  Then go to Step 4.

4. Is:

   - SQL CONNECT TO issued?

     If the referenced server is a connected server, go to Step 5.

     If the server is not a connected server, send the request to the local Db2 subsystem.

   - SQL CONNECT TO not issued?

     If a default server was specified in the EDAPARMS file, go to Step 5.

     Otherwise, send the request to the local database.

5. Is the requested option:

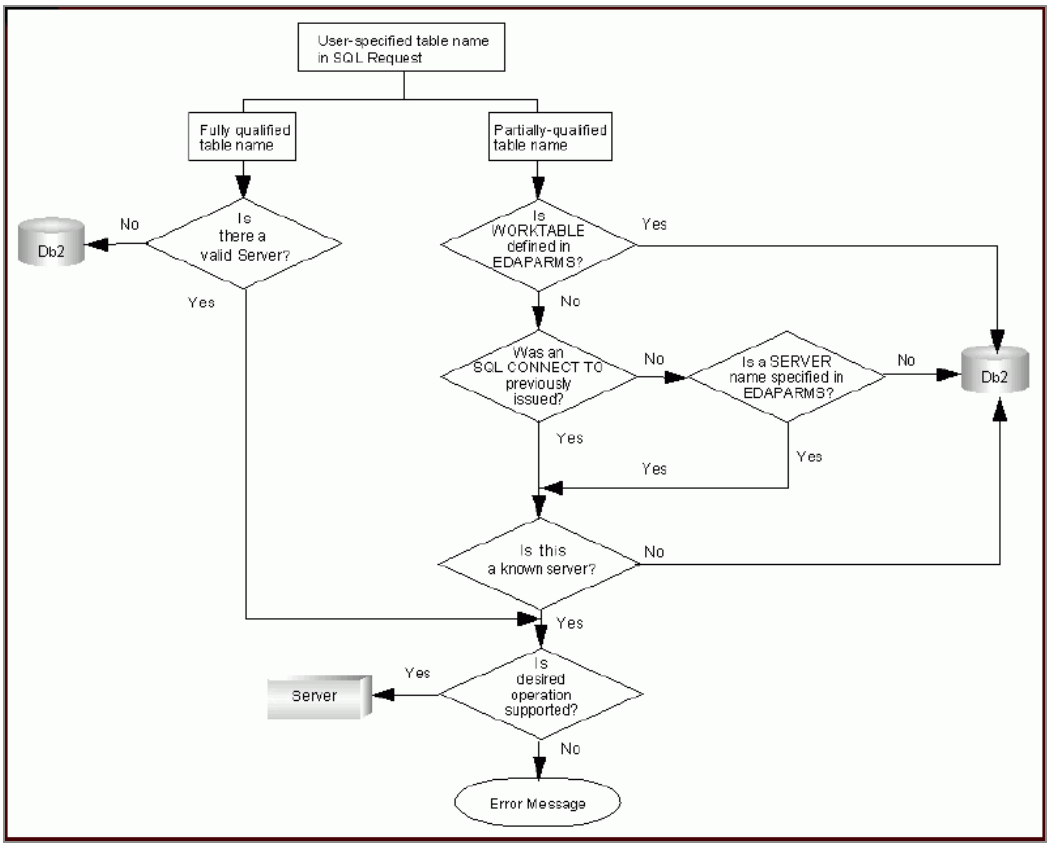
   - Supported by the server?

     Then send the request to the server.

   - Not supported by the server?

     Then the Extender for Db2 supplies the appropriate error message.

This flowchart illustrates how the table destination is determined.



Step 3. Constructing and Dispatching Requests

Once it determines where the SQL request is to be dispatched, the Extender for Db2 constructs a request, depending on whether the request is destined for the local Db2 subsystem or the server.

- For a local Db2 request, it dispatches the original request to the Db2 DSNALI or DSNHLI load module.
- For a server request, it builds server calls around the request and establishes a communications dialog with the server.

Step 4. Returning Data to the Application

When data is returned to the Extender for Db2 from the local Db2 subsystem or the server, the Extender for Db2 populates the appropriate fields in the SQL Data Area (SQLDA) to

contain the data, and the SQL Common Area (SQLCA) to contain the error messages and return codes for communication with the application program.

For data returning from the local Db2 subsystem:

- All retrieved data is returned to the appropriate data area in the SQLDA, as in a typical local Db2 subsystem operation.
- All the local Db2 subsystem error messages and return codes are returned in the SQLCA, as in a typical local Db2 subsystem operation.

For data returning from the server, the Extender for Db2:

- Receives a block of data from the API from any single SQL request.
- Builds an answer set descriptor in the IBM SQLDA format and returns the data back to the application program, as if the application was communicating with the local Db2 subsystem.
- Transfers and communicates all server error messages to the application via the SQLCA.
    - All server error messages and codes are mapped to equivalent local Db2 subsystem error messages and codes.
    - Server messages and codes that have no Db2 equivalent are flagged using the default error number.
    - Non-Db2 mappable server messages cause the Extender for Db2 to return the generic (default) SQLCODE.

For more information about how the Extender for Db2 returns error messages, see Using the ibi WebFOCUS Extender for Db2 and Db2 SQL.

Installing the ibi WebFOCUS Extender for Db2 on z/OS

This section describes how to install the Extender for Db2 in a z/OS environment.

Installation Requirements

This section lists the hardware, software, and system requirements for the Extender for Db2 in a z/OS® environment.

Hardware Requirements

To operate the Extender for Db2, the following hardware is required.

- An IBM® or IBM-compatible mainframe supporting z/OS 2.1 or higher operating systems.
- A minimum of 240 Cylinders of 3390 DASD device for the *qualif*.HOME.LOAD data set containing the Extender for Db2 interface load modules, the IBM Db2 module entry-points, the Extender for Db2 main module, and other Extender modules.

Software Requirements

To operate the Extender for Db2, the following minimum software levels are required.

- Server for z/OS, Version 7.7 or higher
- Db2 v11 or higher
- z/OS 2.1 or higher

When using the Extender for Db2 with a Db2 application product, see the documentation of that product for any additional requirements.

Disk Requirements

The following data sets are needed for the installation of the Extender for Db2.

Data Set	Cylinders of 3380 or 3390
`qualif.HOME.LOAD`	240
`qualif.HOME.DATA`	1

Note: If a PDS Deployment z/OS Server is installed in the same LPAR, you can use the existing qualif.HOME.LOAD library for the Extender for Db2.

System Requirements

Before you install the Extender for Db2, you should have access to a server, with all related data adapters and communications files installed. The server is not required for the Extender for Db2 installation procedure. However, for the Extender to function as a working client, communications between the Extender and the server are essential. For more information about setting up communications, see the *ibi™ WebFOCUS® Reporting Server Administration* manual.

You must install the Distributed Data Facility (DDF). Otherwise, the QMF Draw functionality (which checks the CSECT for the presence of DDF) fails. For more information on whether you have DDF installed with your Db2 subsystem, see your system administrator.

Pre-Installation Issues

The Extender for Db2 is situated between your application (such as QMF) and the local Db2 subsystem (and the server). By providing a Db2 interface to the application, all requests from the application are intercepted by the Extender for Db2 before reaching either the local Db2 subsystem, or the server. The Extender for Db2 uses either one of two standard Db2 Attachment Facilities as the Db2 interface to your application:

- Call Attach Facility
- DSN Command Processor

Before installing the Extender for Db2, you must determine which facility is used by your application and is suitable for your application environment. For more information about the advantages and disadvantages of each facility, see the *IBM Db2 Application Programming and SQL Guide*.

Call Attach Facility

If your application uses the Db2 Call Attach Facility (CAF), there are two ways to access the CAF language interface:

- Explicit LOAD
- Link-edit of DSNALI

The advantages and limitations of using either CAF access mode are fully explained in the *IBM Db2 Application Programming and SQL Guide*. The Extender for Db2 supports both modes of CAF access to Db2. Depending on whether your application uses CAF explicit loading or CAF link-editing of DSNALI, you can install the Extender for Db2 to support either mode of CAF access to Db2.

Explicit LOAD

You can use the explicit loading feature of Db2 with the Extender for Db2. For the Extender, simply allocate the **qualif**.HOMEEXT.LOAD library before the standard Db2 load libraries.

The modules in **qualif**.HOMEEXT.LOAD must be invoked before the modules in the standard Db2 load library, DSNxxx.DSNLOAD because the library **qualif**.HOMEEXT.LOAD contains the entry points DSNALI, DSNHLI, and DSNTIAR, which have similar names to the standard Db2 entry points. If the library **qualif**.HOMEEXT.LOAD is placed before the standard Db2 load library in your JCL or CLIST, the Extender for Db2 module is invoked before the identical modules in the Db2 standard library.

Your application product allocations (such as those for QMF) are usually placed in sequence order between the Extender for Db2 libraries and the standard Db2 load library. For example, QMF requires ISPF load libraries. These are allocated between the Extender for Db2 libraries and the Db2 load library. For more information about the specific Extender for Db2 installation steps, see Installation Procedure.

To allocate the Extender for Db2 libraries for run-time, place the library **qualif**.HOMEEXT.LOAD the application-specific load libraries, and the standard Db2 load libraries, in that order (referred to as the server *concatenation*), in your load library search path. Most installations offer the following choices:

- Create a log in procedure and allocate the server concatenation to STEPLIB.
- For applications invoked under ISPF, allocate the server concatenation to ISPLLIB (if you are not permitted to allocate to STEPLIB).
- Allocate the server concatenation to any other ddname for which the application expects to find the standard Db2 load library allocation.

Note: If the chosen load library file allocation contains APF authorized data sets, the Extender for Db2 load data sets must also be APF authorized. If the Extender for Db2 load data sets is not APF authorized and is integrated with other APF authorized data sets under a ddname such as STEPLIB, the APF authorization of all the data sets is invalidated.

Link-edit of DSNALI

If your application link-edits DSNALI using the CAF interface to Db2, you must instead link-edit the Extender for Db2's DSNALI interceptor stub (in **qualif**.HOMEEXT.LOAD) to your application. For more information about the specific Extender for Db2 installation steps, see Installation Procedure.

DSN Command Processor Facility

The Extender for Db2 supports applications that communicate to Db2 via the DSN Command Processor Facility. As with the support for CAF link-edit with DSNALI, you can enable the Extender for Db2 to use the DSN Command Processor for your application. To do so, simply perform a link-editing step of the Extender for Db2's DSNELI with your application program. For more information about the specific Extender for Db2 installation steps, see Installation Procedure.

Installation Worksheet

The following is an overview of the steps that are performed to install the Extender for Db2, with variations based on specific user and system requirements. The worksheet that follows the overview helps you determine which steps are required for your installation.

Overview

1. Step 1. Unload Two Data Sets From the Tape
2. Step 2. Copy Extender Entry Points
3. Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional)
4. Step 4. Configure the Parameter File EDAPARMS (Optional)
5. Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional)
6. Step 6. Prepare Run-time Allocation Streams

> **Note:** For more information on the software and hardware requirements that must be met before proceeding with installation, see Installation Requirements.

Worksheet

Answer the questions on the following Worksheet, which is intended to help you to differentiate between installation types based on specific user and system requirements. Proceed as instructed, following the detailed information in the referenced steps, as appropriate for your installation.

The Extender for Db2 is shipped as part of the z/OS PDS Deployment Server.

Are you planning to install the PDS Deployment Server in the same z/OS LPAR as the Extender for Db2?

- If Yes:
  - Install the server first, then skip Step 1. Unload Two Data Sets From the Tape and proceed to Step 2. Copy Extender Entry Points . (Note that Step 1 is not necessary because **qualif**.HOME.LOAD and **qualif**.HOME.DATA are unloaded during the PDS Deployment installation.)
  - Use the PDS Deployment EDAENV dataset, at qualif.PDS.server_type.DATA (rather than the simplified one provided in the sample job streams included in

this document).

- If No, run Step 1. Unload Two Data Sets From the Tape, followed by Step 2. Copy Extender Entry Points .

Will Db2 Extender talk directly to the local Db2 subsystem (without using a server)?

- If Yes, run Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional).
- If No, skip Step 3.

Are you installing Db2 Extender without a local Db2 subsystem?

- If Yes, run Step 4. Configure the Parameter File EDAPARMS (Optional) to define a default server name.
- If No, Step 4 is optional.

Does your application require explicit link-edit to IBM's Db2 interface modules (DSNALI, DSNELI or DSNTIAR) or does it explicitly load the interface modules?

- If explicit link-edit, run Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional).
- If explicit load, skip Step 5.

In which mode does your application run?

Interactive: Are you running QMF?

- If Yes, prepare your run-time CLIST from the sample provided in Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI.
- If No, are you running Call Attach Facility (DSNALI) or DSN Command Processor (DSNELI)?
    - If DSNALI, prepare your run-time CLIST by converting the sample JCL in Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI.
    - If DSNELI, prepare your run-time CLIST by converting the sample JCL in Step 6.3. DSN Command Processor (TSO Attach) into a CLIST.

BATCH: Are you running QMF?

- If Yes, prepare your run-time JCL from the sample provided in Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI.
- If No, are you running Call Attach Facility (DSNALI) or DSN Command Processor

(DSNELI)?

- If DSNALI, prepare your run-time JCL from the sample provided in Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI.
- If DSNELI, prepare your run-time JCL from the sample provided in Step 6.3. DSN Command Processor (TSO Attach).

Installation Procedure

Refer to your answers on the installation worksheet, then proceed through the appropriate flow of steps for your installation.

Step 1. Unload Two Data Sets From the Tape

Run an IEBCOPY or allocate and initialize your **qualif**.HOME.DATA and **qualif**.HOME.LOAD datasets.

qualif.HOME.DATA contains the JCL procedures needed for the Extender for Db2 installation.

The JCL is

```
//COPYEM  EXEC PGM=IEBCOPY
//IN1     DD   DISP=(OLD,PASS),DSN=HOME.DATA,LABEL=(1,SL),
//             UNIT=CART,VOL=(,RETAIN,,SER=tapvol)
//IN2     DD   DISP=(OLD,PASS),DSN=HOME.LOAD,LABEL=(12,SL),
//             UNIT=CART,VOL=(,RETAIN,,SER=tapvol)
//OUT1    DD   DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(5,2,20)),
//             UNIT=SYSDA,DSN=qualif.HOME.DATA,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600),
//OUT2    DD   DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(200,20)),
//             UNIT=SYSDA,DSN=qualif.HOME.LOAD,
//             DCB=(RECFM=U,LRECL=0,BLKSIZE=13030),DSNTYPE=LIBRARY
//SYSUT1  DD   UNIT=workunit,SPACE=(CYL,(5,1))
//SYSPRINT DD SYSOUT=*
//SYSIN1 DD *
COPY OUTDD=OUT1,INDD=IN1
COPY OUTDD=OUT2,INDD=IN2
/*
```

where:

workunit

Is the unit for the work data set.

qualif

Is the high-level qualifier for HOME.DATA.

UNIT

Specifies the unit type of the tape drive being used. CART is the default value, but other common names include 3480, TAPE, 3420, 3490.

tapvol

Is the volser label of the installation tape.

After this job has run, *qualif*.HOME.DATA is allocated, cataloged, and populated with the procedures and jobs needed to install the Extender for Db2.

Step 2. Copy Extender Entry Points

This step copies and renames Db2 Extender entry points creating a user library named qualif.HOMEEXT.LOAD. This user library must be concatenated ahead of the standard Db2 load library when running your application.

Edit **qualif**.HOME.DATA(EXTINST2), replacing **qualif** for the appropriate high-level qualifier. Add a job card and submit the job.

```
//*        Job Card Goes Here
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*Purpose: To copy and rename DB2 Extender entry points to a user  *
//*         library. This user library must be concatenated ahead   *
//*         of the standard DB2 load library when running your      *
//*         application.                                            *
//*                                                                 *
//*Substitutions:                                                  *
//*      qualif  should be replaced with high level qualifier for   *
//*              DB2 Extender datasets.                             *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//EXTINST2  EXEC PGM=IEBCOPY
//SYSUT1    DD  SPACE=(CYL,(5,1)),UNIT=SYSDA
```

```
//SYSUT4    DD  SPACE=(CYL,(1,1)),UNIT=SYSDA
//IN01      DD  DISP=SHR,
//          DSN=qualif.HOME.LOAD
//OU01      DD  DISP=(MOD,CATLG),SPACE=(CYL,(5,2)),
//          DSN=qualif.HOMEEXT.LOAD,
//          LIKE=qualif.HOME.LOAD
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  COPYMOD INDD=((IN01,R)),OUTDD=OU01
  SELECT MEMBER=((@DSNALI,DSNALI),(@DSNCLI,DSNCLI),(@DSNELI,DSNELI),   -
  (@DSNTIAR,DSNTIAR),(@DSNHLI,DSNHLI),(@DSNHLI2,DSNHLI2))
/*
```

Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional)

This step links the main Extender for Db2 module with Db2 entry points DSNALI, DSNHLI, DSNHLI2, and DSNTIAR. This is required to enable calls by the Extender for Db2 module to Db2 only for those applications that communicate with the local Db2 subsystem. To perform this step, modify and run *qualif*.HOME.DATA(EXTLINK).

> **Note:** If your application does not need to access the local Db2 subsystem, see Installing the ibi WebFOCUS Extender for Db2 Without Db2.

A sample of *qualif*.HOME.DATA(EXTLINK) follows:

```
//*        Job Card Goes Here
//
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*Purpose: To link DB2 Extender with DB2 Entry points.             *
//*         This is an optional task, only required if you run      *
//*         applications that communicate to the local DB2 system   *
//*         directly (without using the server).                     *
//*                                                                 *
//*Substitutions:                                                   *
//*      qualif  High level qualifier for DB2 Extender dataset.     *
//*      db2hlq  High level qualifier for DB2 Load Library          *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//LKED    EXEC PGM=IEWL,PARM='XREF'
00003700
```

```
//SYSLMOD  DD  DISP=SHR,DSN=qualif.HOMEEXT.LOAD
00003800
//SYSLIB   DD  DISP=SHR,DSN=db2hlq.SDSNLOAD
00003900
//SYSPRINT DD  SYSOUT=*
00004000
//SYSUT1   DD  SPACE=(1024,(50,50)),UNIT=SYSDA
00005100
//SYSLIN   DD  *
00005200
   INCLUDE SYSLIB(DSNTIAR)
00005300
   MODE    AMODE(31),RMODE(ANY)
00005400
   ENTRY   DSNTIAR
00005500
   NAME    QXQTIAR(R)
00005600
   INCLUDE SYSLIB(DSNALI)
00005701
   MODE    AMODE(31),RMODE(ANY)
00005800
   ENTRY   DSNALI
00005901
   NAME    QXQALI(R)
00006001
   INCLUDE SYSLIB(DSNALI)
00006101
   MODE    AMODE(31),RMODE(ANY)
00006201
   ENTRY   DSNHLI
00006301
   NAME    QXQHLI(R)
00006401
   INCLUDE SYSLIB(DSNELI)
00006501
   MODE    AMODE(31),RMODE(ANY)
00006601
   ENTRY   DSNHLI
00006701
   NAME    QXQELI(R)
00006801
/*
00006901
```

Step 4. Configure the Parameter File EDAPARMS (Optional)

Configuring the EDAPARMS parameter file is optional because the Extender for Db2 runs without it. However, you can use the EDAPARMS parameter file to set certain parameters—such as default error numbers, default servers, and continental decimal notation (CDN)—and the destination of partially-qualified names.

If your application loads the Db2 entry points dynamically, this step is not required.

To set up the EDAPARMS parameter file, create a data set named *qualif*.EDAPARMS. For more information about configuring an EDAPARMS file, see Configuring the EDAPARMS File.

Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional)

Your Db2 application may require a static link to the standard Db2 entry points, such as DSNALI and DSNTIAR. If you must link-edit your application with the Extender for Db2's interceptors (DSNALI, DSNELI, and DSNTIAR) instead of Db2 entry points, run the JCL in Step 5.1. Link-Edit JCL to Link Extender for Db2 Interceptors With Your Application (Optional).

Note: If you are using the Call Attach Facility with the explicit load of DSNALI, skip this step and proceed to Step 6. Prepare Run-time Allocation Streams.

Step 5.1. Link-Edit JCL to Link Extender for Db2 Interceptors With Your Application (Optional)

After linking the Extender for Db2 modules to your application program, edit the following sample JCL member qualif.HOME.DATA(EXTLNKAP). This job stream replaces the calls to the IBM modules with calls to the Extender for Db2 modules.

- If your application uses DSNALI as the entry point to interface to Db2, substitute

DSNALI for the first INCLUDE statement.

- If your application uses DSNELI as the entry point to interface to Db2, substitute DSNELI for the first INCLUDE statement.

A sample JCL member *qualif.*HOME.DATA(EXTLNKAP) follows:

```
//*******************************************************************
//* Purpose: Link the user program with DB2 Extender
//*
//* Substitutions:-Change "qualif" into the high level qualifier
//*                for your DB2 Extender datasets.
//*               -Change "userpgm" to the name of your program.
//*               -Change "userentry" to the entry point of your code.
//*
//* If your application uses DSNELI as the entry point to interface to
//* DB2, substitute DSNELI for the first INCLUDE statement.
//*******************************************************************
//LKED     EXEC PGM=IEWL,PARM='LIST,MAP,XREF,LET'
//SYSLMOD  DD   DISP=SHR,DSN=userhlq.LOAD
//EDAEXT   DD   DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//SYSUT1   DD   UNIT=SYSDA,SPACE=(800,(150,50))
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   *
  INCLUDE EDAEXT(DSNALI)
  INCLUDE EDAEXT(DSNTIAR)
  INCLUDE SYSLMOD(userpgm)              <- name of program
  MODE    AMODE(31),RMODE(ANY)
  ENTRY   userentry                     <- program entry point
  NAME  userpgm(R)                    <- name of program
/*
```

where:

userhlq

Is the user high-level qualifier.

qualif

Is the high-level qualifier for your data sets.

userpgm

Is the name of your program.

userentry

Is the entry point of your code.

Step 6. Prepare Run-time Allocation Streams

If using the Call Attach Facility (CAF) with Explicit Load of DSNALI, follow the instructions in Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI to prepare your run-time allocation.

If using the Call Attach Facility (CAF) with Link-edit of DSNALI, follow the instructions in Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI to prepare your run-time allocation.

If using the DSN Command Processor (TSO Attach), follow the instructions in Step 6.3. DSN Command Processor (TSO Attach) to prepare your run-time allocation.

Step 6.1. Call Attach Facility (CAF) With Explicit Load of DSNALI

A sample CLIST and JCL are shown below for CAF explicit load of the DSNALI. To enable your application to invoke the Extender for Db2, you must tailor your log in environment or batch address space to the following allocation streams.

In addition, you must have a correctly configured communications configuration file (CLNTCS3).

In the following examples, *qualif* is the high-level qualifier for your data sets. Other variables shown in the examples are site-dependent.

Sample CLIST

The following is an example of the CLIST necessary to deploy your client using a communicating server with CAF explicit load.

A sample qualif.HOME.DATA(EXTCQMF) follows:

```
/*-------------------------  REXX  -------------------------------*
 * Purpose: Sample CLIST to run QMF with the Extender             *
 *                                                                *
 * Pass the following parameters at invocation:                   *
 *      qualif   High level qualifier for DB2 Extender datasets.  *
 *      db2hlq   High level qualifier for DB2 Libraries.          *
 *      qmfhlq   High level qualifier for QMF Libraries.          *
```

```
*      dbss     DB2 Subsytem name.                              *

*-----------------------------------------------------------------*/
parse upper arg qualif db2hlq qmfhlq dbss
"ALLOC FI(DSQLLIB) DA('"qualif".HOMEEXT.LOAD'," ,
                     "'"qualif".HOME.LOAD'," ,
                     "'ISP.SISPLOAD'," ,
                     "'"qmfhlq".SDSQLOAD'," ,
                     "'"db2hlq".SDSNEXIT'," ,
                     "'"db2hlq".SDSNLOAD') SHR REUSE"
"ALLOC FI(ADMCDATA) DA('GDDM.SADMCDA') SHR REUSE"
"ALLOC FI(ADMDEFS)  DA('GDDM.ADMDEFS') SHR REUSE"
"ALLOC FI(ADMGDF)   DA('GDDM.SADMGDF') SHR REUSE"
"ALLOC FI(ADMSYMBL) DA('GDDM.SADMSYM') SHR REUSE"
"ALLOC FI(ADMCFORM) DA('"qmfhlq".SDSQCHRT') SHR REUSE"
"ALLOC FI(ADMGGMAP) DA('"qmfhlq".SDSQMAPE') SHR REUSE"
"ALLOC FI(DSQPNLE)  DA('"qmfhlq".DSQPNLE')  SHR REUSE"
```

```
"ALLOC FI(DSQEDIT) NEW UNIT(SYSALLDA) CYL SPACE(1 1)
       DSORG(PS) RECFM(F B A) LRECL(79) BLKSIZE(4029)"
```

```
"ALLOC FI(DSQDEBUG) DA(*)                SHR REUSE"
"ALLOC FI(DSQPRINT) DA(*)                SHR REUSE"
"ALLOC F(EDADPDS) DUMMY SHR"
```

```
           /* copy EDAENV contents from sample EXTJQMF jcl */
"ALLOC F(EDAENV)   DA('"qualif".EDAENV')   SHR REUSE"
```

```
           /* copy EDACS3 contents from sample EXTJQMF jcl */
"ALLOC F(EDACS3)   DA('"qualif".EDACS3')   SHR REUSE"
```

```
/* "ALLOC F(EDAPARMS) DA('"qualif".EDAPARMS') SHR REUSE" */
             /* copy IBITRACE contents from sample EXTJQMF jcl */
"ALLOC F(IBITRACE) DA('"qualif".IBITRACE') SHR REUSE"
"ALLOC F(FSTRACE) SYSOUT(X) RECFM(F) LRECL(132) BLKSIZE(132)"
```

```
"ALTLIB ACT APPL(CLIST) DA('"qmfhlq".SDSQCLTE')"
"ALTLIB ACT APPL(EXEC)  DA('"qmfhlq".SDSQEXCE')"
address ispexec "LIBDEF ISPLLIB LIBRARY ID(DSQLLIB) STACK"
```

```
address ispexec "LIBDEF ISPMLIB DATASET ID('"qmfhlq".SDSQMLBE') STACK"
address ispexec "LIBDEF ISPPLIB DATASET ID('"qmfhlq".SDSQPLBE') STACK"
address ispexec "LIBDEF ISPSLIB DATASET ID('"qmfhlq".SDSQSLBE') STACK"
```

```
                    /* QMF invocation */
address ispexec "SELECT PGM(DSQQMFE) NEWAPPL(DSQE) PASSLIB NOCHECK
                 SCRNAME(QMF) PARM(DSQSSUBS="dbss")"
```

```
address ispexec "LIBDEF ISPLLIB"
address ispexec "LIBDEF ISPMLIB"
address ispexec "LIBDEF ISPPLIB"
address ispexec "LIBDEF ISPSLIB"
"ALTLIB DEACT APPL(EXEC)"
"ALTLIB DEACT APPL(CLIST)"
"FREE FI(ADMCDATA ADMCFORM ADMGDF   ADMDEFS  ADMSYMBL ADMGGMAP)"
"FREE FI(DSQDEBUG DSQEDIT  DSQLLIB  DSQPNLE  DSQPRINT EDADPDS)"
"FREE FI(EDAENV   EDACS3   EDAPARMS IBITRACE FSTRACE)"
```

Sample JCL

The following is an example, supplied at **qualif**.HOME.DATA(EXTJQMF), of the JCL necessary to deploy your client using a communicating server with CAF explicit load of the DSNALI.

```
//*        Job Card Goes Here
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*Purpose: Sample JCL to run a QMF procedure in Batch.             *
//*                                                                 *
//*Substitutions:                                                  *
//*      qualif   High level qualifier for DB2 Extender datasets.   *
//*      db2hlq   High level qualifier for DB2 Libraries.           *
//*      qmfhlq   High level qualifier for QMF Libraries.           *
//*      hostn    Server's Host name or Server's IP address.        *
//*      portn    TCP/IP Port number server is listening on.        *
//*      userid   Owner of QMF procedure to be executed.            *
//*      qmfprocs QMF Procedure name.                               *
//*      dbss     DB2 Subsytem name.                                *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//          SET DB2REL=db2hlq
//          SET QMFREL=qmfhlq
```

```
//*************************************************************
//QMFBAT   EXEC  PGM=IKJEFT01,DYNAMNBR=30,TIME=1440,REGION=4096K
//STEPLIB  DD    DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//         DD    DISP=SHR,DSN=qualif.HOME.LOAD
//         DD    DISP=SHR,DSN=&QMFREL..SDSQLOAD
//         DD    DISP=SHR,DSN=&DB2REL..SDSNEXIT
//         DD    DISP=SHR,DSN=&DB2REL..SDSNLOAD
//         DD    DISP=SHR,DSN=ISP.SISPLOAD
//*-------------------------------------------------------------
//*                Extender Client Configuration File
//*-------------------------------------------------------------
//EDACS3   DD    *
NAME        = Client Odin File
NODE = EDASERVE
  BEGIN
    PROTOCOL  = TCP
    CLASS     = CLIENT
    HOST      = hostn    ;Server's Host name or IP address
    PORT      = portn    ;Port # server is listening on
;   TRACE     = 31
  END
/*
```

```
//*-------------------------------------------------------------
//*                Extender Environment
//*-------------------------------------------------------------
   //EDAENV   DD    *
EDACONF=/PDS
FSTRACE=DD:FSTRACE
/*
//EDADPDS  DD    DUMMY
//*-------------------------------------------------------------
//*                Extender EDAPARMS File (Optional)
//*        Extender Traces are enabled in EDAPARMS DD and output
//*             goes to DD QXTRACE (dynamically allocated)
//*-------------------------------------------------------------
//*EDAPARMS  DD  DISP=SHR,DSN=qualif.EDAPARMS
//*-------------------------------------------------------------
//*            API Tracing (trace output goes to DD FSTRACE)
//*-------------------------------------------------------------
//IBITRACE DD *
  SET TRACEON=ALL
//FSTRACE  DD    SYSOUT=*,DCB=(LRECL=132,RECFM=FB,BLKSIZE=132)
//*------------------------------------------------------------
//*               Client Application Allocations
//*------------------------------------------------------------
```

```
//*-------------- TSO Datasets (Required for QMF) ---------------
//SYSPROC  DD  DSN=&QMFREL..SDSQCLTE,DISP=SHR
//SYSEXEC  DD  DSN=&QMFREL..SDSQEXCE,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//*-------------- ISPF DATASETS (REQUIRED FOR QMF) --------------
//ISPPLIB  DD  DSN=&QMFREL..SDSQPLBE,DISP=SHR
//         DD  DSN=ISP.SISPPENU,DISP=SHR
//ISPMLIB  DD  DSN=&QMFREL..SDSQMLBE,DISP=SHR
//         DD  DSN=ISP.SISPMENU,DISP=SHR
//ISPSLIB  DD  DSN=&QMFREL..SDSQSLBE,DISP=SHR
//         DD  DSN=ISP.SISPSENU,DISP=SHR
//ISPTLIB  DD  DSN=ISP.SISPTENU,DISP=SHR
//ISPPROF  DD  UNIT=SYSDA,SPACE=(TRK,(9,1,4)),
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB,DSORG=PO)
//*-------------- QMF Datasets ----------------------------------
//ADMGGMAP  DD  DISP=SHR,DSN=&QMFREL..SDSQMAPE
//ADMCFORM  DD  DISP=SHR,DSN=&QMFREL..SDSQCHRT
//ADMDEFS   DD  DISP=SHR,DSN=CSDDBS.QMF.ADMDEFS
//*-------------- Datasets used by QMF --------------------------
//DSQPRINT   DD  SYSOUT=*
//DSQDEBUG   DD  SYSOUT=*
//*DSQUDUMP  DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
//*SYSUDUMP  DD  SYSOUT=*
//DSQSPILL  DD  DSN=&&SPILL,DISP=(NEW,DELETE),
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
//DSQEDIT   DD  UNIT=SYSDA,SPACE=(TRK,(10,1,5))
```

```
//*-------------------------------------------------------
//*              Executes QMF Batch Procedures
//*-------------------------------------------------------
//SYSTSIN   DD  *
ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) -
         PARM(M=B,I=userid.qmfprocs,S=dbss)
/*
```

Step 6.2. Call Attach Facility (CAF) With Link-Edit of DSNALI

Use this option if your application program requires a link-edit of the standard IBM Db2 Interface modules DSNALI and/or DSNTIAR (instead of the execution of a load macro) to call its routines.

The Extender for Db2 interface modules have the same aliases as the standard IBM modules DSNALI or DSNTIAR. If the IBM modules have been linked into your application program, then these modules must be substituted by the Extender for Db2 interface modules DSNALI or DSNTIAR residing in the *qualif*.HOMEEXT.LOAD.

- If your application accesses both a server and the local Db2 subsystem, your execution job stream must allocate the load libraries in sequence order ahead of the Db2 standard load libraries. The allocation is done in STEPLIB.

  Step 6.2 Example 1:

  ```
  //RUNSTEP EXEC PGM=userprogram
  //STEPLIB DD   DISP=SHR,DSN=qualif.HOMEEXT.LOAD
  //        DD   DISP=SHR,DSN=qualif.HOME.LOAD
  //        DD   DISP=SHR,DSN=DSN810.SDSNEXIT
  //        DD   DISP=SHR,DSN=DSN810.SDSNLOAD
  ```

- If your application does not access the local Db2 subsystem, then you only need the Db2 Extender load libraries in STEPLIB.

  Step 6.2 Example 2:

  ```
  //RUNSTEP EXEC PGM=userprogram//STEPLIB DD
  DISP=SHR,DSN=qualif.HOMEEXT.LOAD
  //        DD   DISP=SHR,DSN=qualif.HOME.LOAD
  ```

  where:

  userpgm

  Is the name of your program.

  qualif

  Is the high-level qualifier for your data sets.

The following is a full Job stream that runs a program with Db2 Extender using the Link-Edit facility of DSNALI (CAF); it is provided at **qualif**.HOME.DATA(EXTCBCAF). (Notice that it uses the case described in example 1.)

```
//*        Job Card Goes Here
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//*Purpose:Sample JCL to run CAF (DSNALI) program with DB2 Extender *
//*                                                                 *
```

```
//*Substitutions:                                                    *
//*      qualif   High level qualifier for DB2 Extender datasets.    *
//*      db2hlq   High level qualifier for DB2 Libraries.            *
//*      hostn    Server's Host name or Server's IP address.         *
//*      portn    TCP/IP Port number server is listening on.         *
//*      userlib  Dataset where XTDCOB program resides.              *
//*      XTDCOB   replace with the name of your program              *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//         SET   DB2REL=db2hlq
//CAFRUN   EXEC  PGM=XTDCOB
//STEPLIB  DD    DISP=SHR,DSN=userlib.LOAD
//         DD    DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//         DD    DISP=SHR,DSN=qualif.HOME.LOAD
//         DD    DISP=SHR,DSN=&DB2REL..SDSNEXIT
//         DD    DISP=SHR,DSN=&DB2REL..SDSNLOAD
//*-----------------------------------------------------------------
//*                Extender Client Configuration File
//*-----------------------------------------------------------------
//EDACS3   DD    *
NAME        = Client Odin File
NODE = EDASERVE
  BEGIN
    PROTOCOL  = TCP
    CLASS     = CLIENT
    HOST      = hostn    ;Server's Host name or IP address
    PORT      = portn    ;Port # server is listening on
;   TRACE     = 31
 END
/*
```

```
//*-----------------------------------------------------------------
//*                Extender Environment
//*-----------------------------------------------------------------
//EDAENV   DD    *
FSTRACE=DD:FSTRACE
EDACONF=/PDS
/*
//EDADPDS  DD    DUMMY
//*-----------------------------------------------------------------
//*                Extender EDAPARMS File (Optional)
//*        Extender Traces are enabled in EDAPARMS DD and output
//*             goes to DD QXTRACE (dynamically allocated)
//*-----------------------------------------------------------------
//*EDAPARMS  DD  DISP=SHR,DSN=qualif.EDAPARMS
//*-----------------------------------------------------------------
```

```
//*            API Tracing (trace output goes to DD FSTRACE)
//*-----------------------------------------------------------------
//IBITRACE DD *
   SET TRACEON=ALL
//FSTRACE  DD    SYSOUT=*,DCB=(LRECL=132,RECFM=FB,BLKSIZE=132)
//*-------------------------------------------------------------
//*               User Application Allocations
//*-------------------------------------------------------------
//XTDPRM    DD  *
DEBUG=N,BATCH=Y
/*
//DBGOUT  DD    SYSOUT=*
//SYSOUT  DD    SYSOUT=*
//SYSIN   DD  *
SQL
SELECT COUNTRY,CAR,MODEL,BODYTYPE FROM EDASERVE.ANYNAME.CAR
END
SQL
SELECT LAST_NAME,FIRST_NAME FROM EDASERVE.ANYNAME.EMPLOYEE
END
EXIT
/*
```

Step 6.3. DSN Command Processor (TSO Attach)

The Extender for Db2 provides a back end to your application and contains invocations similar to the Db2 DSN Command Processor invocations.

- Application requests destined to the local Db2 subsystem are first intercepted by the Extender for Db2, which then passes them through to the DSN Command Processor Facility of the local Db2 subsystem.
- Application requests destined for server databases are routed to the server.

In all cases, the application invokes the Extender using the calls and conventions of the DSN Command Processor Facility.

To enable your application directly to the Extender for Db2's interface modules, instead of to the standard Db2 DSN Command Processor modules, you must link-edit the Extender for Db2 interface modules to your application. After you perform this link-edit, your application invokes the Extender for Db2's DSN interface, which input mimics the standard Db2 Command Processor Facility.

The following is an example, provided at *qualif*.HOME.DATA(EXTCBTSO), of the run-time allocation job stream for the client using a communicating client/server architecture with DSN.

```
//*        Job Card Goes Here
//*
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
//*Purpose:Sample JCL to run TSO (DSNELI) program with DB2 Extender*
//*                                                                *
//*Substitutions:                                                 *
//*      qualif   High level qualifier for DB2 Extender datasets.  *
//*      db2hlq   High level qualifier for DB2 Libraries.          *
//*      dbss     DB2 Subsytem name.                               *
//*      hostn    Server's Host name or Server's IP address.       *
//*      portn    TCP/IP Port number server is listening on.       *
//*      userlib  Dataset where TSOCOB program resides.            *
//*      TSOCOB   replace with the name of your program.           *
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

//         SET   DB2REL=db2hlq
//TSORUN   EXEC  PGM=IKJEFT01
//STEPLIB  DD    DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//         DD    DISP=SHR,DSN=qualif.HOME.LOAD
//         DD    DISP=SHR,DSN=&DB2REL..SDSNEXIT
//         DD    DISP=SHR,DSN=&DB2REL..SDSNLOAD
```

```
//*-----------------------------------------------------------------
//*                Extender Client Configuration File
//*-----------------------------------------------------------------
//EDACS3   DD    *
NAME        = Client Odin File
NODE = EDASERVE
  BEGIN
    PROTOCOL  = TCP
    CLASS     = CLIENT
    HOST      = hostn    ;Server's Host name or IP address
    PORT      = portn    ;Port # server is listening on
;   TRACE     = 31
  END
/*
//*-----------------------------------------------------------------
//*                Extender Environment
//*-----------------------------------------------------------------
//EDAENV   DD    *
FSTRACE=DD:FSTRACE
EDACONF=/PDS
```

```
/*
//EDADPDS  DD    DUMMY
//*-----------------------------------------------------------------
//*                Extender EDAPARMS File (Optional)
//*        Extender Traces are enabled in EDAPARMS DD and output
//*             goes to DD QXTRACE (dynamically allocated)
//*-----------------------------------------------------------------
//*EDAPARMS  DD  DISP=SHR,DSN=qualif.EDAPARMS
//*-----------------------------------------------------------------
//*            API Tracing (trace output goes to DD FSTRACE)
//*-----------------------------------------------------------------
//IBITRACE DD *
  SET TRACEON=ALL
//FSTRACE  DD    SYSOUT=*,DCB=(LRECL=132,RECFM=FB,BLKSIZE=132)
```

```
//*-----------------------------------------------------------------
//*               User Application Allocations
//*-----------------------------------------------------------------
//XTDPRM    DD  *
DEBUG=N,BATCH=Y
/*
//DBGOUT  DD    SYSOUT=*
//SYSOUT  DD    SYSOUT=*
//SYSTSPRT  DD  SYSOUT=*
//SYSTSIN   DD  *
  DSN SYSTEM(dbss)
  RUN PROGRAM (TSOCOB)      -
      LIB('userlib.LOAD')
  END
/*
//SYSIN   DD  *
SQL
SELECT COUNTRY,CAR,MODEL,BODYTYPE FROM EDASERVE.ANYNAME.CAR
END
SQL
SELECT LAST_NAME,FIRST_NAME FROM EDASERVE.ANYNAME.EMPLOYEE
END
EXIT
/*
```

Installing the ibi WebFOCUS Extender for Db2 Without Db2

You can install the Extender for Db2 without Db2. Db2 front-end applications can then issue server requests and query non-Db2 and other relational and non-relational databases without the presence of a Db2 database. An example is an installation site is a requirement of querying IMS and VSAM data. The Extender for Db2 enables access through a Db2 application (such as QMF), but does not specifically require the installation to have Db2.

To install the Extender for Db2 without Db2, perform the steps described in Installation Procedure with the following changes:

1. Unload two data sets from the tape.
2. Copy Extender Entry points.
3. Omit Step 3. Link-edit the Extender for Db2 Module With Db2 Entry Points (Optional). This step enables the Extender for Db2 to communicate with Db2, which is not present.
4. Set up the parameter file EDAPARMS. Set the following parameters in this file:

   ```
   EDASERVER  = edaserver_name
   DBMS       = EDA
   ```

5. Link the Extender for Db2 interface modules with your application program (optional). Complete Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional) only if your Db2 application is statically linked to the Db2 standard IBM modules, such as DSNALI, DSNELI, DSNTIAR. Running this step links your application to the Extender for Db2 interceptor modules (also called DSNALI, DSNELI, DSNTIAR, and so on) instead of Db2's.
   - Tailor and run the JCL in Step 5.1. Link-Edit JCL to Link Extender for Db2 Interceptors With Your Application (Optional).
6. Customize the sample CLIST/JCLs provided for the user program.
7. Run your application.

Configuring the EDAPARMS File

This section describes how to set up the EDAPARMS configuration file.

Overview of the EDAPARMS File

The EDAPARMS file is a sequential file that configures the Extender for Db2 for different products and can also contain installation-specific parameters. The file is created using the system editor and cannot be altered at run-time.

With the EDAPARMS file, you can:

- Define product work tables or partially-qualified table names. Table names appearing in the EDAPARMS file direct the Extender for Db2 to dispatch SQL requests to the local database subsystem.
- Define a generic system-wide error message for a server return error that is other than the default.

  ```
  "-901  System Error Has Occurred"
  ```

- Define a default server for partially-qualified table names. If no server was specified in an SQL CONNECT TO request, the default server name is used. If an SQL CONNECT RESET was issued, then this default server is referenced.
- Enable traces for Db2 Extender.

Use of the EDAPARMS file is optional. It can be omitted from an installation without adversely affecting the operation of the Extender for Db2.

Creating the EDAPARMS File

To set up the EDAPARMS parameter file, create a data set named *qualif*.EDAPARMS

```
* EDAPARMS file
   ERRNUM    =  errornum
   EDASERVER =  defserver
   DBMS      =  database_type
   DECIMAL   =  decimal_point_type
   DATE      =  date_format
   ENABLE    = trace_level
   WORKTABLE =  worktablename
      .
      .
      .
   WORKTABLE =  worktablename
```

where:

*****

Denotes comments within the EDAPARMS file. An asterisk (*) in column one identifies the line as a comment. More than one comment is possible within the EDAPARMS file.

ERRNUM

Represents an Extender for Db2 error number. The value supplied must be an integer within the range -2147483647 to 2147483648. (-901) is the default value. The Extender for Db2 uses it for server error codes that cannot be mapped to Db2 error codes. It is recommended that you assign a negative three- or four-digit number for simplicity. Avoid using a valid Db2 error code.

If more than one ERRNUM is defined, the last definition is used.

EDASERVER

Is the default server for unresolved partially qualified tables. Verify that this server is defined in both client and server communications configuration files. This field can be upto eight bytes in length.

If more than one EDASERVER is defined, the last definition is used.

If the EDASERVER keyword references a server not defined in the communications configuration file, an error message is produced.

DBMS

There are three valid values for this parameter.

Value	Description
`BOTH`	Enables the Extender to access both the local database subsystem and databases residing on the server. BOTH is the default value.
`NATIVE`	Enables the Extender to access only the local database subsystem and prohibits access to databases residing on the server.
`EDA`	Enables the Extender to access only databases residing on the server and prohibits access to the local database subsystem.

If more than one DBMS is defined, the last definition is used.

DECIMAL

Enables you to specify whether you want a period or a comma for the decimal point in a numeric value. PERIOD is the default value. If you specify the value COMMA here, it allows either a period or a comma to be used as the decimal point.

If more than one DECIMAL is defined, the last definition is used.

DATE

Enables you to specify the output format of a date field. There are four valid values for this parameter.

Value	Description
`ISO`	The output format is YYYY-MM-DD. For example, the output could be: 1992-05-21. ISO is the default value.
`USA`	The output format is MM/DD/YYYY. For example, the output could be: 05/21/1992.
`EUR`	The output format is DD.MM.YYYY. For example, the output could be: 21.05.1992.

Value	Description
`JIS`	The output format is YYYY-MM-DD. For example, the output could be: 1992-05-21.

If more than one DATE is defined, the last definition is used.

ENABLE

Enables several levels of tracing for Db2 Extender. API traces are written to DDNAME FSTRACE and all other traces are written to DDNAME QXTRACE (dynamically allocated when needed). Possible trace values are:

Value	Description
`MESSAGES`	Corresponds to the EDALOG file used in earlier releases. Users can record all warning and error messages originating from the server or the Extender for Db2. EDALOG is useful if a display or output of warnings and messages is desired. This trace is especially useful for Db2 application products that do not use the DSNTIAR message formatting facility.
`SQLTRACE`	Corresponds to the QXSQLTRC file used in earlier releases. Displays the SQL statement types (NATIVE or EDA). In addition, users can see how a statement string is physically prepared for dispatching.
`RDITRACE`	This is the main component trace for Db2 Extender. It shows that the inputs received, the processing of SQLDA, and the proper handling of every query and its results.
`PRSTRACE`	SQL Parser output. Displays command type (for example, SELECT, INSERT, UPDATE), number of tables, and table names.

Value	Description
`APITRACEx`	Where x is from 1 to 9 API trace levels. It can be enabled separately for levels 1 to 9.

WORKTABLE

Is the name of a product worktable (for example, Q.PROFILE). The Extender directs all requests referencing these worktables to Db2. More than one WORKTABLE keyword definition can exist in the EDAPARMS file. For more information on naming conventions and worktables, see Using the ibi WebFOCUS Extender for Db2 and SQL.

Using the EDAPARMS File

The following guidelines are helpful when using the EDAPARMS file.

- Assign an ERRNUM if an application-specific error code is desired for all server errors that cannot be mapped to the local Db2 subsystem error code set.
- Assign an EDASERVER for the default destination for requests containing partially-qualified table names (two-part names) to be a specific server. If EDASERVER is omitted, the local Db2 subsystem is the default destination for requests containing partially-qualified table names. When an EDASERVER is assigned, it can be overridden by using an explicit SQL CONNECT TO verb to connect to the desired server for partially-qualified table names. The EDASERVER keyword is for default, unresolved destinations to assign to requests containing partially-qualified table names. For more information, see Connecting to Multiple ibi WebFOCUS Reporting Servers.
- Assign WORKTABLEs if you have already assigned an EDASERVER, but want to send requests containing specific partially-qualified table names to the local Db2 subsystem. WORKTABLEs can be product-specific partially-qualified table names that reside in the local Db2 subsystem from a product installation. Essentially, the WORKTABLE keyword overrides the effects of an EDASERVER assignment for partially-qualified table names for product-specific tables. You can assign a WORKTABLE without an EDASERVER assignment present. However, this is not necessary, as the default destination for partially-qualified tables is the local Db2 subsystem.
- To allocate the EDAPARMS file when using the Extender for Db2, add the following statement to the client CLIST

```
ALLOC F(EDAPARMS) DA('qualif.EDAPARMS') SHR
```

or add the following statement to the client JCL:

```
//EDAPARMS     DD   DSN=qualif.EDAPARMS,DISP=SHR
```

Using the ibi WebFOCUS Extender for Db2 for Db2 Administrative Operations

This section describes how to use security and tracing with the Extender for Db2 in the z/OS environment.

Security

Security is implemented at the server level. The Extender for Db2 provides an explicit means to access the database engine by supporting an EDA IMMEDIATE command. This command allows an SQL user to:

- Issue directives to the server identifying the user and password.
- Validate the user to a server security subsystem.
- Verify entry to specific adapters.

Using the EDA IMMEDIATE Command

The syntax of the EDA IMMEDIATE command is

```
EDA [server] command
```

where:

server

Is optional. If included, it is the name of a valid server defined in the client communications configuration file. A value for this parameter must be supplied to override the default server name defined in the EDAPARMS file or if the SQL CONNECT command is not used.

command

Is a valid EDA IMMEDIATE command as described below.

The proper usage of this command is to assign it to a host variable string, and issue an EXECUTE IMMEDIATE macro on the host variable string, as illustrated in the following example.

```
string = "EDA server command"
EXEC SQL
EXECUTE IMMEDIATE :string
END-EXEC
```

Supported Types of Security

There are three types of client-supported security. To enable each type of security, use the EDA IMMEDIATE command. For more information, see Using the EDA IMMEDIATE Command.

- **General server user ID/password.** Used to provide a user ID and password to be validated by the server. When using this type of security, the following command must be run before any other SQL command is sent to the server.

  ```
  string="EDA edaserver  SET EDAUSER=id  EDAPASS=password"
  EXEC SQL
  EXECUTE IMMEDIATE :string
  END-EXEC
  ```

  This command does not initiate communication with the server.

- **Server DBA password.** Used to set the DBA password as the general server password. This option provides file-level security. When using this type of security, issue the following command.

  ```
  string = "EDA edaserver SET PASS = dbapassword"
  EXEC SQL
  EXECUTE IMMEDIATE :string
  END-EXEC
  ```

  For detailed information, see *Providing Data Source Security: DBA* in the *Describing Data With ibi™ WebFOCUS® Language* manual.

- **Adapter-specific password security.** Used with adapters that need specific passwords. For example, the format for setting a log in string directed at an adapter interfacing with a Teradata database is:

```
string="EDA edaserver SQL SQLDBC SET DBCLOGON tdpid/userid,pwd;"
EXEC SQL
EXECUTE IMMEDIATE string
END-EXEC
```

In the previous string the value of the command is:

```
SQL SQLDBC SET DBCLOGON tdpid/userid,pwd;
```

For security administration, the command can be set to any literal as required by the specific adapter.

Client Security

If your server security is enabled, the client or the Extender for Db2 must identify itself with a user ID and password, to access the server. For more information, see Supported Types of Security. For example, in z/OS/QMF, the user would type in query mode .

```
EDA edaserver SET EDAUSER=userid EDAPASS=password
```

where:

edaserver

Is a valid server name, as identified in the client configuration file as partner_lu_name.

userid

Is a valid user ID known and acceptable to the server.

password

Is a valid password known and acceptable to the server.

In QMF, the user does not need to specify EXECUTE IMMEDIATE since this is done by QMF internally. For other Db2 applications, check to confirm that EXECUTE IMMEDIATE is implicitly performed; otherwise, you may need to specify it explicitly. For more information, see Security.

Tracing

You can enable two types of traces:

- **Extender traces.**To enable these traces, see the instructions under the EDAPARMS file (as described in Configuring the EDAPARMS File).
- **API traces.** To enable these traces, in addition to enabling traces under EDAPARMS you must allocate the following DDNAMES:

DDNAME	Description
`FSTRACE`	Allocate FSTRACE with the following DCB parameters: `RECFM=FB LRECL=132 BLKSIZE=13200`
`IBITRACE`	IBITRACE is a one line file containing the following command: `SET TRACEON=ALL`
`EDAENV`	Make sure the following line is coded: `FSTRACE=DD:FSTRACE`

Note: You can allocate these three files permanently to the jcl streams since traces will only be produced if they are enabled from EDAPARMS file.

QXUDUMP Data Set File Allocation

With QXUDUMP, users can produce a dump data set containing the results from an Extender for Db2 abend. Users requiring assistance to interpret the contents of this dump should contact Customer Support Services.

The following DCB characteristics for the CLIST or JCL allocation for QXUDUMP for a 3380 device are:

```
RECFM=VBA
LRECL=125
BLKSIZE=882
```

The CLIST allocation is:

```
ALLOC F(QXUDUMP) DA('qualif.QXUDUMP') MOD -
RECFM(VBA) LRECL(125) BLKSIZE(882)
```

The corresponding JCL allocation for QXUDUMP is:

```
//QXUDUMP     DD   DSN=qualif.QXUXTDUMP,DISP=SHR
                   DCB=(RECFM=VBA,LRECL=125,BLKSIZE=882)
```

Error Message Formatting Facility (DSNTIAR)

DSNTIAR is a message formatting facility feature of Db2. A typical Db2 application invokes DSNTIAR to convert an SQLCODE to a text message only if DSNTIAR is statically link-edited with the application program.

The Extender for Db2 also supports a DSNTIAR entry point. This feature enables conversion of both the SQLCODEs from Db2 and the return codes from the server to meaningful text messages.

Using the ibi WebFOCUS Extender for Db2 and SQL

This section describes Translation Services and how the Extender for Db2 compares to ANSI SQL.

Table Naming Conventions

Users must follow table naming conventions when accessing tables. SQL requests must be structured to contain either:

- All tables are accessible on one server.
- All tables are accessible under the local or default Db2 subsystem.

Mixed destination requests containing both local Db2 subsystem tables and database tables accessible to the server are not permitted.

The Extender for Db2 directs each SQL SELECT statement issued against a table based on the set of rules outlined in this section. There are three types of tables known to the user.

- Fully-Qualified Tables
- Partially-Qualified Tables
- Product Work Tables

Fully-Qualified Tables

Fully-qualified table names consist of three-part names and contain an explicit location. The syntax for a fully-qualified table is

```
location.creator.tablename
```

where:

location

Is the location of the data. If the location is one of the current servers in the Client communications configuration file, then the location is a valid server location. This is an 8-byte (character) field.

creator

Is the creator of the table. This is an 8-byte (character) field. Currently, it is ignored for tables residing in the server.

tablename

Is the name of the table. For the server, it must be a Master file name or a system-defined catalog name.

For example,

```
EDASERVE.JAMES.EMPLOYEE
```

Indicates that the table EMPLOYEE was created by JAMES and can be accessed through a server called EDASERVE. The name of the server is determined at installation. The server's communications configuration file contains the name of the server and is established at installation.

Partially-Qualified Tables

Partially-qualified table names have one- or two-part names and do not contain an explicit location. For partially-qualified table names, the Extender for Db2 parses the SQL CONNECT TO server command to determine whether to route the request directly to Db2, or to the server. Partially-qualified table names can be listed in the EDAPARMS configuration file under the WORKTABLE keyword.

The syntax for a partially-qualified table is

```
creator.tablename
```

where:

creator

Is the creator of the table. This is an 8-byte (character) field. It is ignored for tables residing in the server.

tablename

Is the name of the table. For the server, it must be a Master file name or system-defined catalog name.

For example,

```
WATSON.SUPPLIER
```

Indicates that the SUPPLIER table was created by WATSON.

While parsing the SQL request, the Extender for Db2 determines if the object has a one-part or two-part name. If so, the Extender for Db2 handles the object as a partially-qualified table. Based on the most recent SQL CONNECT TO issue, the Extender for Db2 resolves the destination of the partially-qualified table name to be the current server.

- If an SQL CONNECT TO is issued to a valid server, the request is sent to the server. Otherwise, it is sent to Db2.
- If no SQL CONNECT TO was issued, the Extender for Db2 checks the EDAPARMS file to determine if a default server was declared using the EDASERVER keyword. For information about the contents of the EDAPARMS file, see Configuring the EDAPARMS File.
    - If a default server is declared in EDAPARMS, the request goes to that default server.

      However, if the partially-qualified table name matches a worktable specified in the EDAPARMS file, the default server is overridden and the request is sent to the local Db2 subsystem.
    - If there is no default server and no worktables are defined, the request is sent to Db2. For more information on support for the SQL CONNECT verb, see Connecting to Multiple ibi WebFOCUS Reporting Servers.

Product Work Tables

Each product that uses the Extender for Db2 can have its own work tables and profile logs, all located outside of the server. They must be listed in the WORKTABLE keyword list in the

EDAPARMS configuration file. The Extender for Db2 considers all of them non-server and sends corresponding queries referencing these tables to Db2.

Table names should be 8-byte (character) fields. The rules in Partially-Qualified Tables also apply to naming product work tables.

SQL Translation

This section describes how the Extender for Db2 and the SQL Translator access heterogeneous relational and non-relational databases. The topics include:

- Column Name Resolution
- Alternate Column Names
- Dynamically Defined Virtual Fields
- Answer Set Generation
- Additional Features for SQL Translation Services
- SQL Translation Services Limitations

Column Name Resolution

When resolving column names, the SQL Translator does not accept a unique truncation of a column name as a valid name for that column. For example, if you had a table with a column named EMPID, you cannot refer to that column as EMP (assuming no other column referenced in the request began with those three letters). You must refer to the column by its full name, EMPID.

Alternate Column Names

The SQL Translator is fully ANSI compliant; therefore, the user cannot use ALIAS= facility in the Master File. The ANSI specification states that a column has only one name. To rename a column logically, the application should create a view of that table with a different column name.

Dynamically Defined Virtual Fields

It is not possible to define a virtual field dynamically in a remote procedure and use it in subsequent SQL statements against that table. Virtual fields must be defined in the Master File. For more information about Master Files, consult the *ibi™ WebFOCUS® Adapter Administration* manual.

Answer Set Generation

To provide completely transparent SQL access, SQL Translation Services create a Cartesian product style answer set in all cases, regardless of the nature of the underlying DBMS. A Cartesian product style answer set is in keeping with the SQL-based nature of the data access mechanism.

What Is a Cartesian Product?

A Cartesian product or set multiplication is defined as the pairing of each element of x with every element of y. This type of response is the expected result of a relational JOIN.

What Does This Mean to You?

This means that in some instances, the answer sets received are larger than expected. The Cartesian product generation only affects situations involving JOINs, either implicit or explicit.

- **Explicit join.** Defined in the SQL statement used to generate the answer set.
- **Implicit join.** Reference to any data structure made up of independent parts, such as segments in a hierarchy.

For example, there is a three-segment hierarchical database. The top segment represents departments, one child segment represents employees, and the other—the furniture used by that department. Assume that the payroll department has 20 employees and 22 desks. If you ask for all of the employees and furniture from the payroll department, the SQL user would expect to get each employee listed 22 times, once for each desk. This type of answer is the Cartesian product set answer, and is consistent with the result you would expect from an SQL-based DBMS, such as Db2.

The Cartesian product set answer can appear only under certain specific circumstances, as in the above example, with multi-path requests in a hierarchical data structure. In general, it only results in a repetition of rows.

Answer Set Generation Logic

The algorithm used to interpret the generation of answer sets is straightforward. This algorithm is the structure around which answer set generation is performed. This algorithm will be familiar to any experienced SQL user, but may be a new experience to application developers and users that are more familiar with other DBMS systems.

This algorithm does not correspond to the internal mechanism of generating answer sets, but is a convenient means of thinking about that process. The internal mechanisms are different because they have been optimized for performance in specific DBMS environments.

The (simplified) algorithm is:

1. Create the Cartesian product of every logical table referenced in the answer set. A logical table is defined as:
    - A relational table or view.
    - A flat file (VSAM, C-ISAM, etc.).
    - A segment of a hierarchical database.
    - A segment of a network database.
    - Any other data structure designated as a segment in the Master File.
2. Remove all rows from the Cartesian product that do not pass the screening criteria specified in the WHERE clause of the SQL statement.
3. Calculate any valued expressions in the SQL statement.
4. Perform the ordering and grouping specified in the SQL statement.
5. Remove any repeated values if specified with the DISTINCT operator.
6. Calculate the results of any column functions (SUM, COUNT).
7. Remove the result rows that do not correspond to screening conditions in the HAVING clause of the SQL statement.
8. Return the answer set.

What to Look For

Three things help to explain the Cartesian product answer set generation, particularly for users unfamiliar with SQL-based DBMSs.

- Results of aggregate functions, such as sum or count, are generated after the Cartesian product is created. This means that in the department/employee/furniture example, if the user requests a sum of the employees' salaries in every department that had enough desks for all employees, they receive the result of 22 times the sum of the salaries for the payroll department. This is because each employee is associated with each desk, and each desk with each employee, resulting in 440 (20 X 22) items in the Cartesian product instead of 20.
- The Cartesian product is generated for the referenced logical tables in the request. For a hierarchical database, this refers to the referenced subtree. If the department/employee/furniture request is changed to ask for only a sum of the salaries of the employees, with no reference to the desks, the result is the expected sum of the salaries. Since the desks were not referenced, the employees would not be repeated in the personnel department.
- A row is created only when every logical table in the join exists. If you ask for the sum of salaries where there are enough desks, you do not get a result for departments that did not own any desks. This behavior is usually referred to as an *inner join*.

Additional Features for SQL Translation Services

- **ANSI Level 2.**The SQL Translator is compliant with the ANSI Level 2 SQL definition.
- **Virtual Column Support.**The SQL Translator supports the definition of virtual column in the Master File, and the use of these in any capacity in which you would use a regular database column.
- **SQL Join Improvements.**The SQL Translator handles virtually any join predicate based on an equality condition, regardless of the indexing or other characteristic of the column. This eliminates the necessity to understand any of the characteristics of the DBMS in which the data is stored.
- **View Creation/Deletion.**You can CREATE and DROP temporary views in server databases.
- **SQL Translation Performance Enhancements.**The SQL Translator provides

improved functionality and performance. A sophisticated JOIN optimizer is included.

These and other features allow server users to develop the client/server applications quickly and easily.

SQL Translation Services Limitations

The following limitations exist in the Extender for Db2.

1. A maximum of 16 tables may be referenced in a single SQL statement.
2. A maximum of six SELECT statements may be joined by the UNION operator.
3. Correlated subqueries are not supported, except for Db2 tables.
4. The maximum number of columns that may be in the column list of a SELECT statement is 256. Your actual limit for a given query may be less, because the SQL Translation Services may reserve several of these items for its own use.
5. Date and time arithmetic are not supported.
6. The maximum number of fields in a GROUP BY or ORDER BY clause is 32. Again, the SQL Translation Services may reserve a small number of these for its own use.
7. The maximum length of an SQL statement is 4000 bytes.
8. The maximum size of a row is 32K.

Using the ibi WebFOCUS Extender for Db2 and Db2 SQL

This section describes enhancements and limitations of the Extender for Db2 when working with Db2 SQL.

General Considerations

Db2 applications enabled with the Extender for Db2 find differences between server and Db2 SQL. These differences involve syntax, semantics, and error message reporting. The Extender for Db2 user must be aware of these differences while preparing a Db2 application, to take full advantage of server capabilities and to abide by its limitations. This chapter describes the differences between server and Db2 SQL, so that Extender for Db2 users can utilize the interface effectively.

As a rule, all SQL must follow the ANSI Level 2 standard. The server encompasses the ANSI Level 2 standard, plus some Db2 extensions. However, not all extensions are supported.

These notes pertain to the following table, which lists SQL considerations.

- Tables A and B are fully-qualified server tables.
- ESRV*x* are specific server location names.
- C*x* specifies a column name.
- T*x* specifies a correlation tag.

Description	Examples	Messages trace
Joins between two tables residing under different servers are not possible.	**Valid:** `SELECT T1.C1,`	

Description	Examples	Messages trace
	`T2.C2 FROM ESRV1.X.A T1, ESRV1.X.B T2 WHERE T1.C1 = T2.C2;`	
	Invalid: `SELECT T1.C1,T2.C2 FROM ESRV1.X.A T1, ESRV2.X.B T2 WHERE T1.C1=T2.C2`	Cannot refer to multiple servers.
Joins between a local Db2 subsystem (see note) with a server-accessible table is not supported.	**Valid:** `SELECT T1.C1 FROM ESRV.X.TABL T1, ESRV.DB2.TABL T2 WHERE T1.C1=T2.C2;`	
	Invalid: `SELECT T1.C1 FROM DB2.TABL T1, ESRV1.X.TABL T2 WHERE T1.C1=T2.C2;`	None. The user receives SQLCODE=-512 from Db2.
GROUP BY fields must be referred to by name, not by positional value.	**Valid:** `SELECT C1,C2, C3 FROM A GROUP BY C1,C2,C3`	

Description	Examples	Messages trace

	Invalid:
	`SELECT C1, C2, C3`
	`FROM A`
	`GROUP BY 1,2,3`
		FOC14069 SQL syntax error.

Correlated subqueries supported only for Db2 tables.	**Valid:**
	`SELECT DISTINCT`
	`T1.C1,`
	`T1.C2`
	`FROM A T1, B T2`
	`WHERE T1.C1 <`
	`'00100'`
	`AND`
	`T1.C2 =`
	`T2.C2 AND`
	`T2.C3 =`
	`'EDA';`
	Valid:
	`SELECT C1,C2`
	`FROM A`
	`WHERE C1 < '00100'`
	`AND`
	`C2 IN (SELECT C2`
	`FROM B`
	`WHERE C3 =`
	`'EDA');`	

	Invalid:
	`SELECT T1.C1,`
	`T1.C2`
	`FROM A T1`
	`WHERE T1.C1 <`
	`'00100'`
		EDA14013
		UNSUPPORT-ED SYNTAX:
		Correlated Subquery SQL CODE=-84.
		Description Examples

Description	Examples	Messages trace

	`AND EXISTS (SELECT 1 FROM B WHERE C2 = T1.C2 AND C3 = 'EDA');`
	
	`(valid query if both A and B are DB2 tables defined on the EDA/SERVER)`
		EDALOG Message.

The maximum number of columns per table is 256. (The actual limit for a given query may be less, because the SQL Translation Services may reserve several of these items for its own use).	**Valid:** `SELECT A.name, A.name...(250 times) FROM A`	
	Invalid: `SELECT A.name, A.name...(750 times) FROM A`	EDA00005 THE NUMBER OF VERB OBJECTS EXCEEDS THE MAXIMUM, which maps to SQLCODE=-840. Too many columns in a request.

Note:

- Local Db2 subsystem refers to the Db2 subsystem normally accessed via dynamic SQL by a Db2 application independent of the server. Usually, two-part table names are used for this type of access. To access server data, three-part names are used, with the first part referencing a valid server location. By default, the Extender for Db2 enables access to both the local Db2 subsystem and the server data in separate requests using the respective naming conventions, but does not support the mixing of naming conventions where the location names are different in the same request.
- The maximum supported size of a decimal column is 31-digit decimal columns.
- If data types are equal, arithmetical operations return values in the same data type of the operands, otherwise a FLOAT(15,3) is returned. If you observe unexpected overflow indicators in your data, make sure that the USAGE specification in your server's Master File for the table is sufficiently large for the data type being displayed.

Data Conversion

This section describes supported conversions of data returning from a request to data described in an SQL Data Area (SQLDA). Users must verify that field types returned by a request from a table are compatible with field types designated in the SQLDA. The application communicating with the Extender for Db2 is assumed to receive the data from the SQLDA.

The data retrieved by the adapter is converted to a server format. The Extender for Db2 then converts the data from the server format to Db2 format and returns the data in a standard SQLDA. The rules for converting data retrieved by the adapter are described in the *ibi™ WebFOCUS® Adapter Administration* manual. The rules for converting data in a server format to Db2 format are described in this section.

In general, format conversion between similar types (such as numeric to numeric and character to character) is preferred and, when appropriate, is performed. This is illustrated in the following table.

Server Format	Db2 Format
Numeric	INTEGER, SMALLINT, FLOAT, DECIMAL
Alphanumeric	VARCHAR, CHAR
Zoned	DECIMAL
Date	ISO, USA, EUR, JIS

Note: Db2 normally overwrites, truncates, or does not access the data area. The Extender for Db2 operates in a similar capacity.

Error Handling

Error messages originating from the server are converted to an appropriate Db2 error code. This Db2 error code is then returned to the application via the SQLCA. Applications communicating with the Extender for Db2 are assumed to receive error messages and tokens via the SQLCA.

Error codes returned to a client application by the Extender for Db2 are communicated using the SQLCA. There are three sources that can generate error messages.

- Extender for Db2—internal messages from the Extender for Db2.
- Server API—status codes (for example -9, -12).
- Server—internal messages from the server (for example EDA251, EDA757).

The Extender for Db2 attempts to map all messages originating from the server to an appropriate SQLCODE and tokens, and returns this information back to the application via SQLCA's sqlcode and sqlerrmc fields. For more information on the current mapping of the messages, see ibi WebFOCUS Extender for Db2 Error Messages and Codes.

If there is no known mapping to an appropriate Db2 error code, the Extender for Db2 returns a generic system error message. The following message indicates a system error:

```
-901 A SYSTEM ERROR HAS OCCURRED
```

The Db2 code can be overwritten by the ERRNUM keyword in the EDAPARMS file. If users receive the generic error message on the screen, they may allocate an EDALOG DD card on the client address space to resolve the error message further.

Parameter Marker Support

The Extender for Db2 provides limited support of parameter markers with dynamic SQL, as described in the *IBM Db2 Application Programming and SQL Guide*.

Applications using parameter markers must first DECLARE the cursor, issue a CONNECT to the server, PREPARE the dynamic SQL SELECT request, and perform an OPEN cursor using an SQLDA, with the following example format:

```
EXEC SQL OPEN C1 USING DESCRIPTOR SQLDA1
```

Note: Non-SELECTs using EXECUTEs are not supported. Also, host-variables in the USING clause of the OPEN are not supported.

Discrepancies Between the ibi WebFOCUS Reporting Server and Db2 SQL

The server is modeled after the SQL standard as defined by ANSI Level 2 (with some Db2 extensions). Db2 users may find some discrepancies between the server and Db2 SQL. The discrepancies between the server and Db2 SQL are in the specification of requests, and in expecting certain data answer set displays and Db2-like error messages.

These discrepancies are in the following categories.

- Db2 Non-ANSI Compliant SQL Requests.
- Answer Set Displays.
- Db2 Error Codes (SQLCODEs).

To ensure smooth usage and operation, users should be familiar with these discrepancies before using Db2 applications with the Extender for Db2.

Db2 Non-ANSI Compliant SQL Requests

Db2 non-ANSI compliant SQL requests are specific to Db2 SQL and do not conform to ANSI Level 2 SQL. The server follows the ANSI Level 2 SQL standard. The following lists unsupported SQL requests.

1. Blank spaces placed before, in the middle of, or after a table name surrounded by quotation marks are unsupported and are treated as significant by the Extender for Db2, for example:

   ```
   SELECT * FROM "EDASERVE"."X"."TABLENAME   "
   SELECT * FROM "EDASERVE"."X"."   TABLENAME"
   SELECT * FROM "EDASERVE"."X"."TABLE   NAME"
   ```

   These requests return an SQLCODE=-901 with EDALOG message (-901 is the default value for ERRNUM):

   ```
   EDA14063 TABLE NAME CONTAINS ILLEGAL CHARACTER
   ```

2. Invalid expressions used with aggregate functions return an EDALOG message

   ```
   EDA14007 SYNTAX ERROR AFTER...
   ```

   which maps to SQLCODE=-104. Db2 produces an answer set for the following example SQL statement:

   ```
   SELECT AVG(DISTINCT NINTPART/QAVALINV)
   FROM EDASERVE.X.VPARTINV A
   ```

3. VARCHAR, GRAPHIC, DECIMAL, FROM, WHERE, and other standard SQL keywords used as table names in the three-part name, return an EDALOG message

   ```
   EDA14007 SYNTAX ERROR AFTER...
   ```

   which maps to SQLCODE=-104. Examples of SQL statements that generate this error message are:

`SELECT *`
`FROM`
`EDASERVE.X.VARCHAR`
	Db2 returns data if table X.VARCHAR exists.

`SELECT VARGRAPHIC`
`(NINTPART)`
`FROM`
`EDASERVE.X.VPARTINV`
	Db2 returns SQLCODE=-171.

`SELECT TIMESTAMP`
`FROM`
`EDASERVE.X.VPARTINV`
	Db2 returns data.

4. Db2 does not allow the creation of a table with hyphenated column names. The server allows hyphenated column names in the Master File. However, using the Extender for Db2 to query a defined hyphenated column results in the EDALOG message

   ```
   EDA1400 SQLCODE IS -206
   ```

   which is equivalent to:

   ```
   Column not found in the named table
   ```

   This result is correct when querying a Db2 table via the server.

Answer Set Displays

Answer set display discrepancies are comprised of those resultant SQL answer set formats that differ from those normally expected by Db2 users (such as placement of null data value during sorting). Answer set display considerations are:

1. For columns displayed as a result of an arithmetical computation or an aggregate function, column names are generated. These names are SQLDEF01, SQLDEF02, etc. They appear in the SQLDA in the answer set as the column name of the computed

data column.

2. When a sort is executed in ascending order, null data appears at the end of the resulting answer set. In Db2, null values are expected to appear at the top of the answer set. The opposite is true if the sort is executed in descending order.

Db2 Error Codes (SQLCODEs)

In some cases, Db2 error codes (SQLCODEs) generated from server applications differ from those returned by Db2 for similar SQL statements. This is because the server has a different parsing mechanism for SQL requests than Db2, and cannot duplicate some Db2-specific features. In other cases, the server returns an SQLCODE equivalent to that returned by Db2. For more information on cross-referencing SQLCODEs with the expected Db2 codes, see Extender for Db2 Cross-reference Table, which contains examples of SQL statements that generate the Extender for Db2 SQLCODE. SQLCODEs are discussed here.

1. If a column in the ORDER BY list is not contained in the SELECT list, the server returns an SQLCODE=-206 for a column not found, or SQLCODE=-901 for a column which exists for a table. Db2 would return an SQLCODE=-208.

2. When a numeric or date/time column is specified in a LIKE predicate, the server returns an SQLCODE=-132

   ```
   INVALID LIKE PREDICATE
   ```

   whereas Db2 would return an SQLCODE=-414. An example of this type of statement is:

   ```
   SELECT * FROM EDASERVE.X.VENDPART
   WHERE NVENPART LIKE '%700%'
   ```

3. Invalid use of ">" returns SQLCODE=-104, with

   ```
   EDA14007 SYNTAX ERROR AFTER...
   ```

   Db2 returns SQLCODE=-115

   ```
   Invalid use of '>'
   ```

   An example of a statement that produces such an error is:

```
SELECT *
FROM EDASERVE.X.VPARTINV
WHERE NINTPART > ANY QAVALINV
```

4. An exponential value exceeding limits in the SELECT clause returns

```
EDA202 INTERRUPT. FLOATING VALUE OVERFLOW
```

which is mapped to SQLCODE=-802

```
Exception error on arithmetic operation.
```

Db2 returns SQLCODE=-405

```
Numeric literal out of range.
```

An example of an SQL statement that generates this error message is:

```
SELECT (NINTPART - 7.3E75)
FROM EDASERVE.X.VPARTINV A
```

5. An invalid WHERE clause using LIKE "% %" returns an EDALOG message

```
EDA14007 SYNTAX ERROR AFTER...
```

which maps to SQLCODE=-104. Db2 returns SQLCODE=-312

```
Undefined or unusable host variable.
```

An example of an SQL statement that generates this error message is:

```
SELECT DISTINCT *
FROM EDASERVE.X.SYSCOLUM
WHERE NAME LIKE "% %"
```

Enhancements to Db2

The Extender for Db2 provides the following enhancements to Db2. Generally, the server follows the ANSI Level 2 standard. In certain situations, the following server features may

be preferable:

1. Comparing a literal longer than the column definition in the Master File results in an EDALOG message

   ```
   EDA0015 TEST VALUE IS LONGER THAN THE FIELD FORMAT LENGTH
   ```

   which maps to an SQLCODE=-901. Db2 performs the comparison and always returns a false.

   An example of an SQL statement that generates this error is:

   ```
   SELECT *
   FROM EDASERVE.X.VPRODUCT
   WHERE NPRODUCT = 'BBRDDO'
   ```

   where the Master File for the field NPRODUCT is a 4-byte character.

2. If a column function expects a column name and does not find it, but finds an arithmetic computation instead, the server performs the computation and returns the result multiplied by the number of rows in the table. For example,

   ```
   SELECT SUM(10*10)
   ```

   The server returns 3500 as a result for a table with 35 rows (that is, 10*10*35). Db2 returns an SQLCODE=-111

   ```
   Column function does not include a column name.
   ```

   To justify this behavior, Db2 also returns 3500 as a result for a table with 35 rows for SELECT SUM(10 *10 + SEATS - SEATS), where SEATS is a defined column of the table.

3. Creator names that exceed 8 characters generate an error message in Db2. The server accepts creator names greater than 8 characters for some relational databases, other than Db2.

4. Nested aggregate functions, which are not permitted in Db2, return SQLCODE=-112

   ```
   Operand of a column function is another column function or DISTINCT
   followed by an expression.
   ```

   An example of an SQL statement that uses a nested aggregate function is:

```
SELECT AVG(ID*MAX(COMM))
FROM EDASERVE.Q.QSTAFF
```

5. An invalid use of parameter markers returns

```
EDA14041 USING clause has fewer values than ? in statement
```

which maps to SQLCODE=-313

```
Invalid use of ?.
```

Db2 returns the same SQLCODE.

An example of a statement that generates this type of error message is:

```
SELECT * FROM EDASERVE.X.QSTAFF
WHERE NAME = ?
```

6. A repeated GROUP keyword in the SELECT statement returns SQLCODE=-104, whereas Db2 returns SQLCODE=-199. An example of a statement that generates this type of error message is:

```
SELECT MANAGER
FROM EDASERVE.X.QORG
GROUP GROUP BY DIVISION
```

7. A join of greater than five server tables consumes a large amount of CPU power. This is a consequence of any Db2 application with a request of this nature.
8. An SQL statement greater than 256 columns in the ORDER BY clause returns the EDALOG message

```
E300010 THE NUMBER OF SORT FIELDS EXCEEDS THE MAXIMUM
```

and maps to SQLCODE=-136. In Db2 this statement returns an SQLCODE=-136

```
Sort key length is greater than 4000 bytes
```

when the string following the ORDER BY is greater than 4000 bytes.

An example of an SQL statement that generates this error is:

```
SELECT *
FROM EDASERVE.X.VPARTINV
ORDER BY name, name, name   (repeated 4000 times)
```

9. The NAME column in SYSCOLUM is 66 chars; in the Db2 catalog, this NAME column in SYSIBM.SYSCOLUMNS is 18 characters. When using a prompted query in QMF to DESCRIBE the NAME column from SYSCOLUMNS, a view is created where the CNAME column is truncated from 66 chars to 30 chars, which is the maximum in the SQLDA. To create this view, append the following:

```
-INCLUDE EDALONG
-INCLUDE EDAQMFV
```

   at the end of the user profile on the Server.

10. QMF's QBE DRAW of two tables with a WHERE clause generates a query where #$@OUTER is a correlation tag. The QBE-generated query containing #$@OUTER as a correlation tag is presented to the server as a quoted string and the proper response is returned. An example of a QBE-generated SQL statement is:

```
SELECT #$@OUTER."NAME", #$@OUTER."ADDRESS"
FROM "EDASERVE"."X"."EMPLOYEE" #@$OUTER
WHERE #$OUTER."NAME" = 'JAMES'
```

11. Using a built-in function in the WHERE clause, such as WHERE MIN(COLUMN) = 2, returns SQLCODE=-120

```
A WHERE clause includes a column function
```

   which is what Db2 expects.

12. Incompatible columns in the UNION SELECT list returns SQLCODE=-415

```
Corresponding columns of the operands of a UNION do not have
comparable column descriptions
```

   which is what Db2 returns as well.

   The following example returns SQLCODE=415

```
SELECT NAME
```

```
FROM EDASERVE.X.QSTAFF
UNION
SELECT EMPNO
FROM EDASERVE.X.QSTAFF
```

13. Order of precedence using the NOT function in the WHERE clause, such as a statement containing WHERE NOT (A=B AND C=D) OR E=F, is performed according to the ANSI Level 2 standard.
14. QMF Prompted Query generates SELECT SALARY - 7.3E75 that returns in EDALOG

    ```
    EDA0202 INTERRUPT. FLOATING VALUE OVERFLOW
    ```

    which maps to SQLCODE=-802

    ```
    Exception error.
    ```

    Db2 returns SQLCODE=-405

    ```
    There is currently no specific parser detection of a floating value
    underflow.
    ```

15. In QMF Prompt Query LIST? TABLE attempts to query a view DSQEC_TABS_RDB2. A member in EDARPC.DATA, named EDAQMFV, creates this view. To run this CREATE VIEW, place the following at the end of the user profile on the server, and have the FOCEXEC DD card defined with EDARPC.DATA to the server.

    ```
    -INCLUDE EDALONG
    -INCLUDE EDAQMFV
    ```

SAA CPI Functionality Checklist

The IBM Systems Application Architecture (SAA) Common Programming Interface (CPI) Functionality Checklist table, which follows the list of notes, evaluates Db2, the Extender for Db2, and server API support for the Extender for Db2. This table is based on the CPI checklist in the *SAA CPI Database Level 2 Reference Manual* (SC26-4798-00).

These notes pertain to the following table.

- Db2 support is for Version 11 running under z/OS.
- The Extender for Db2 refers to the SQL and associated statements that can be used to access server-controlled tables. The Extender for Db2 supports all Db2 functions for direct access to Db2.
- The server API refers to the SQL and associated statements that use server Translation Services. The Direct Passthru mode enables use of any SQL statement that can be accepted by the DBMS.
- *equivalent* signifies that the server API provides the equivalent function, but not through the same syntax.
- *future* signifies that this functionality is implemented in a future release, but has not been assigned to a particular release level yet.

SQL Feature	Db2	Extender for Db2	Server API
SELECT Expressions			
SELECT list	yes	yes	yes
FROM clause	yes	yes	yes
WHERE clause	yes	yes	yes
GROUP BY clause	yes	yes	yes
HAVING clause	yes	yes	yes
ORDER BY clause	yes	yes	yes
UNION	yes	yes	yes
UNION ALL	yes	yes	yes
Data Definitions			
ALTER TABLE	yes	n/a	no
COMMENT ON	yes	n/a	no
CREATE INDEX	yes	n/a	no
CREATE TABLE	yes	n/a	yes
CREATE VIEW	yes	yes (1)	yes (1)
DROP	yes	n/a	no
Authorizations			

SQL Feature	Db2	Extender for Db2	Server API
GRANT	yes	n/a	no
REVOKE	yes	n/a	no
Basic Statements			
Searched DELETE	yes	yes	no
INSERT	yes	yes	yes
SELECT INTO	yes	yes	no
Searched UPDATE	yes	yes	no
Cursor Operations			
CLOSE	yes	yes	equivalent
DECLARE CURSOR	yes	yes	equivalent
Positioned DELETE	yes	yes	no
FETCH	yes	yes	equivalent
OPEN	yes	yes	equivalent
SELECT ... FOR UPDATE	yes	yes	future
Positioned UPDATE	yes	yes	future
Dynamic Facilities			
DESCRIBE	yes	yes	yes (2)
EXECUTE	yes	yes (3)	yes
EXECUTE IMMEDIATE	yes	yes	equivalent
PREPARE	yes	yes	yes
Connection and Transaction			
CONNECT	yes	yes	yes (4)
COMMIT	yes	yes	yes
ROLLBACK	yes	yes	yes

SQL Feature	Db2	Extender for Db2	Server API
Miscellaneous Statements			
BEGIN DECLARE SECTION	yes	yes	n/a
END DECLARE SECTION	yes	yes	n/a
INCLUDE	yes	yes	n/a
LOCK TABLE	yes	no	no
OPTIMIZE FOR	yes	no	no
WHENEVER	yes	no	n/a
Data Types			
CHARACTER	yes	yes	yes
DATE	yes	yes	yes
DECIMAL	yes	yes	yes
FLOAT (single)	yes	yes	yes
FLOAT (double)	yes	yes	yes
GRAPHIC	yes	yes	yes
INTEGER	yes	yes	yes
NUMERIC	yes	yes	yes
SMALLINT	yes	yes	yes
TIME	yes	yes	yes
TIMESTAMP	yes	yes	yes
VARCHAR	yes	yes	yes
VARCHARAPHIC	yes	yes	yes
Column Functions			
AVG	yes	yes	yes
COUNT	yes	yes	yes
MAX	yes	yes	yes
MIN	yes	yes	yes
SUM	yes	yes	yes
Scalar Function			

SQL Feature	Db2	Extender for Db2	Server API
CHAR	yes	yes	yes
DATE	yes	yes	yes
DAY	yes	yes	yes
DAYS	yes	yes	yes
DECIMAL	yes	yes	yes
FLOAT	yes	yes	yes
HOUR	yes	yes	yes
INTEGER	yes	yes	yes
LENGTH	yes	yes	yes
MICROSECOND	yes	yes	yes
MINUTE	yes	yes	yes
MONTH	yes	yes	yes
SECOND	yes	yes	yes
SUBSTR	yes	yes	yes
TIME	yes	yes	yes
TIMESTAMP	yes	yes	yes
VALUE	yes	yes	yes
VARGRAPHIC	yes	yes	yes
YEAR	yes	yes	yes
Predicates			
ALL/ANY	yes	yes	yes
BETWEEN	yes	yes	yes
EXISTS	yes	yes	yes
IN	yes	yes	yes
IS NULL	yes	yes	yes
LIKE	yes	yes	yes
Other Language Elements			
Arithmetic Operators	yes	yes	yes
Column References	yes	yes	yes
Comparison Operators	yes	yes	yes
Date/Time	yes	yes	yes

SQL Feature	Db2	Extender for Db2	Server API
Arithmetic	yes	no (5)	no (5)
Delimited	yes	yes	equivalent
Identifiers	yes	yes	yes
Foreign Keys	yes	no(6)	no (6)
Host Variable	yes	yes	yes
References	yes	no	no
Indicator	yes	yes	yes
Variables	yes	no	no
Null Values	yes	yes	equivalent
Null Value	yes	yes	equivalent
Arithmetic	yes	yes	equivalent
Ordinary	yes	no	no
Identifiers	yes	yes	yes
Primary Keys			
Search Conditions			
Special Registers			
SQLCA			
SQLDA			
SQLSTATE			
String			
Concatenation			
Host Languages			
Application	yes	yes	yes
Generator	yes	yes	yes
C	yes	yes	yes
COBOL	yes	yes	yes
FORTRAN			
PL/I			
Procedure Languages			
RPG	yes	future	yes

Note:

1. Views in the server system are temporary objects that exist only for the duration of your connection with that server. A view cannot be erased until the client session is terminated.
2. Also, enables the user to DESCRIBE a table without having to PREPARE a SELECT statement.
3. Not all SQL statements can be EXECUTEd. Only a previously PREPAREd SQL statement can be run.
4. The server API enables a user to be connected to multiple servers simultaneously.
5. Host variable references within a SELECT are not supported. Parameter markers are supported.
6. A null value is treated as a zero in an arithmetic computation. For example, 10 plus a null value results in a 10.

Extender for Db2 Cross-reference Table

Extender for Db2 users might receive SQLCODEs in the SQLCA that differ from expected SQLCODEs for similarly constructed SQL statements. To help diagnose the SQL request, the following table cross-references SQLCODEs received via the Extender for Db2 with possible Db2 SQLCODEs returned for equivalent Db2 SQL statements.

The following table also contains examples of SQL statements that generate the Extender for Db2 SQLCODE.

Extender for Db2 SQLCODE	Possible Equivalent Db2 SQLCODE	Type of SQL Statement	See ...
-84	-401	SELECT (USER+1).	Data Conversion
-104	-115	Invalid use of ">" as in WHERE SAL > ANY NUMB.	Enhancements to Db2

Extender for Db2 SQLCODE	Possible Equivalent Db2 SQLCODE	Type of SQL Statement	See ...
	-171	Invalid syntax with VARGRAPHIC scalar function.	General Considerations
	-199	Repeated GROUP keyword in SELECT statement.	Enhancements to Db2
	-312	Invalid LIKE expression in WHERE clause.	Data Conversion
	-904	SQL reserved words used as table names.	Data Conversion
	Data Results	A function has an invalid expression.	Data Conversion
-802	-405	SELECT SALARY-7.3E75.	Data Conversion
-840	Data Results	Number of columns exceed the server limit of 256.	Enhancements to Db2
	-129	Db2 parses the FROM list and gets a limit exceeded; whereas the server parses the SELECT list limit first.	Enhancements to Db2
Data Results	-107	Creator name in table reference is > 8 characters.	Enhancements to Db2
Data Results	-419	A server enhancement performs divide	Enhancements to Db2

Extender for Db2 SQLCODE	Possible Equivalent Db2 SQLCODE	Type of SQL Statement	See ...
		operation which results in a negative scale.	
-206	Not Permitted	Db2 does not permit tables with hyphenated column names.	General Considerations
-901ERRNUM	-105, -110, -115,-130, -159, -164,-207, -208, -414,-537, -553, -554,-603, -614, -815	The server does not support Db2-specific features.	

Using ibi WebFOCUS Extender for Db2 Application Implementations

This section provides examples of using specific third-party applications with the Extender for Db2.

Using QMF

The example in this section illustrates how QMF can access VSAM data using the Extender for Db2, by:

- Presenting the run-time CLIST.
- Displaying an actual QMF query that accesses a VSAM table.
- Displaying the final screen of VSAM data returned to QMF.

Run-time CLIST

This sample run-time CLIST sets the run-time environment for QMF and the Extender for Db2. This CLIST is provided at **qualif**.HOME.DATA(EXTCQMF).

```
/*-------------------------  REXX  -------------------------------*
 * Purpose: Sample CLIST to run QMF with the Extender             *
 *                                                                *
 * Pass the following parameters at invocation:                   *
 *      qualif   High level qualifier for DB2 Extender datasets.  *
 *      db2hlq   High level qualifier for DB2 Libraries.          *
 *      qmfhlq   High level qualifier for QMF Libraries.          *
 *      dbss     DB2 Subsytem name.                               *

 *----------------------------------------------------------------*/
 parse upper arg qualif db2hlq qmfhlq dbss
 "ALLOC FI(DSQLLIB) DA('"qualif".HOMEEXT.LOAD'," ,
```

```
                      ""qualif".HOME.LOAD'," ,
                      "'ISP.SISPLOAD'," ,
                      ""qmfhlq".SDSQLOAD'," ,
                      ""db2hlq".SDSNEXIT'," ,
                  ""db2hlq".SDSNLOAD') SHR REUSE"
"ALLOC FI(ADMCDATA) DA('GDDM.SADMCDA') SHR REUSE"
"ALLOC FI(ADMDEFS) DA('GDDM.ADMDEFS') SHR REUSE"
"ALLOC FI(ADMGDF) DA('GDDM.SADMGDF') SHR REUSE"
"ALLOC FI(ADMSYMBL) DA('GDDM.SADMSYM') SHR REUSE"
"ALLOC FI(ADMCFORM) DA('"qmfhlq".SDSQCHRT') SHR REUSE"
"ALLOC FI(ADMGGMAP) DA('"qmfhlq".SDSQMAPE') SHR REUSE"
"ALLOC FI(DSQPNLE) DA('"qmfhlq".DSQPNLE') SHR REUSE"
"ALLOC FI(DSQEDIT) NEW UNIT(SYSALLDA) CYL SPACE(1 1)
       DSORG(PS) RECFM(F B A) LRECL(79) BLKSIZE(4029)"
"ALLOC FI(DSQDEBUG) DA(*) SHR REUSE"
"ALLOC FI(DSQPRINT) DA(*) SHR REUSE"
"ALLOC F(EDADPDS) DUMMY SHR"
```

```
           /* copy EDAENV contents from sample EXTJQMF jcl */
"ALLOC F(EDAENV)   DA('"qualif".EDAENV')   SHR REUSE"
```

```
           /* copy EDACS3 contents from sample EXTJQMF jcl */
"ALLOC F(EDACS3)   DA('"qualif".EDACS3')   SHR REUSE"
```

```
/* "ALLOC F(EDAPARMS) DA('"qualif".EDAPARMS') SHR REUSE" */
             /* copy IBITRACE contents from sample EXTJQMF jcl */
"ALLOC F(IBITRACE) DA('"qualif".IBITRACE') SHR REUSE"
"ALLOC F(FSTRACE) SYSOUT(X) RECFM(F) LRECL(132) BLKSIZE(132)"
```

```
"ALTLIB ACT APPL(CLIST) DA('"qmfhlq".SDSQCLTE')"
"ALTLIB ACT APPL(EXEC)  DA('"qmfhlq".SDSQEXCE')"
address ispexec "LIBDEF ISPLLIB LIBRARY ID(DSQLLIB) STACK"
address ispexec "LIBDEF ISPMLIB DATASET ID('"qmfhlq".SDSQMLBE') STACK"
address ispexec "LIBDEF ISPPLIB DATASET ID('"qmfhlq".SDSQPLBE') STACK"
address ispexec "LIBDEF ISPSLIB DATASET ID('"qmfhlq".SDSQSLBE') STACK"
```

```
                    /* QMF invocation */
address ispexec "SELECT PGM(DSQQMFE) NEWAPPL(DSQE) PASSLIB NOCHECK
                 SCRNAME(QMF) PARM(DSQSSUBS="dbss")"
```

```
address ispexec "LIBDEF ISPLLIB"
address ispexec "LIBDEF ISPMLIB"
address ispexec "LIBDEF ISPPLIB"
address ispexec "LIBDEF ISPSLIB"
"ALTLIB DEACT APPL(EXEC)"
"ALTLIB DEACT APPL(CLIST)"
"FREE FI(ADMCDATA ADMCFORM ADMGDF   ADMDEFS  ADMSYMBL ADMGGMAP)"
"FREE FI(DSQDEBUG DSQEDIT  DSQLLIB  DSQPNLE  DSQPRINT EDADPDS)"
"FREE FI(EDAENV   EDACS3   EDAPARMS IBITRACE FSTRACE)"
```

The SQL SELECT Statement

This screen displays a simple SQL SELECT statement. This is a query that a user would typically enter in QMF.

```
SQL QUERY                                          LINE 1


SELECT * FROM EDASERVE.X.VPARTINV


*** END ***


1=Help   2=Run    3=End     4=Print     5=Chart       6=Draw
7=Backward  8=Forward  9=Form  10=Insert   11=Delete
12=Report
 OK, QUERY is displayed.

COMMAND ===>
SCROLL ===> PAGE
```

The Data Returned to QMF

This sample QMF screen shows the requested data returned to the client.

```
REPORT                                    LINE 1      POS 1
79


 NINTPART    NAMEPART    QHELDINV     QAVALINV    UNIT
NVENPART
-----------  ----------  -----------  ---------   ---- -------
---
          1  Screw                 0        900     EA
1001
         10  Glue                  0         80     EA
7006
         23  Globe Frame           0         30     EA
8002
        111  Sphere                0         33     EA
7005
        112  Axis Pin              0         40     EA
1005
        113  Wing Nut              0         60     EA
8005
        150  White Board           0         15     EA
4002
        151  Slate Board           0         25     EA
4003
        226  Short Leg             0        160     EA
2002
        227  Tall Leg              0        100     EA
3002
        300  Tray                  0         35     EA
5002
        462  Table Top             0         28     EA
6004
        554  Frame                 0         40     EA
7003

 1=Help     2=       3=End    4=Print       5=Chart
6=Query
 7=Backward  8=Forward  9=Form   10=Left     11=Right
12=
 OK, this is the REPORT from your RUN command.
 COMMAND ===>
 SCROLL ===> PAGE
```

Using the Rocket Compiler for QMF

This section describes how to install the Extender for Db2 with the Rocket Software Compiler for QMF. For more information, see Installing the ibi WebFOCUS Extender for Db2 on z/OS.

Prerequisites

Confirm that your system has sufficient DASD to accommodate both the Rocket Compiler and the Extender for Db2 software. Also, if installing a server, see the appropriate server manual for specific hardware requirements. For more information on DASD memory requirements for the Extender for Db2, see Installing the ibi WebFOCUS Extender for Db2 on z/OS.

Installing the Rocket Compiler for QMF

Install the Rocket Compiler for QMF according to the documentation for Rocket QMF supplied by Rocket Software. For the Rocket Compiler, skip the step that creates a Db2 catalog snapshot of VSAM files. After the Rocket Compiler is installed, use the facility to generate, compile, and link-edit a COBOL program to verify independent functionality.

Installing the Extender for Db2

Users should already have QMF installed and running with Db2 on z/OS with the appropriate release levels. Also, verify that all the client and server components are installed and fully functional. For more information, see the appropriate documentation.

To install the Extender for Db2 client, follow these steps, described in detail in Installing the ibi WebFOCUS Extender for Db2 on z/OS.

1. Allocate disk space for the Extender for Db2 libraries.
2. Unload the distribution tape.
3. Link-edit the main Extender for Db2 module with your Db2 entry points.

> **Note:** Db2 Extender is LE compliant and, therefore, requires all 3GL programs like COBOL to be linked using 31-bit addressing.

`AMODE(31) RMODE(ANY)` for 31-bit addressing

The Extender for Db2 main module is linked with 31-bit addressing mode in *qualif*.HOME.LOAD.

Also, verify that the addressing mode is compatible in the generated Rocket Compiler JCL to compile and link-edit the generated COBOL2 program.

4. Set up the parameter file EDAPARMS (Optional). This step is optional and is dependent on-site preferences and needs.

5. Link the Extender for Db2 statically to the Rocket-generated COBOL2 program. Use a modified version of Step 5. Link the Extender for Db2 Interceptors With Your Application (Optional).

   The Rocket Compiler for QMF generates and compiles QMF report programs in COBOL2, then performs a static link-edit to these programs. To ensure that the generated COBOL2 program can access the Extender for Db2, modify the link-edit step of the Rocket-generated JCL that generates, compiles, and links the Rocket-generated COBOL2 program. Place the Extender for Db2 *qualif*.HOMEEXT.LOAD and *qualif*.HOME.LOAD libraries in SYSLIB of the link-edit step, ahead of the standard Db2 load library, in concatenation sequence:

   ```
   //SYSLIB  DD DISP=SHR,DSN=qualif.HOMEEXT.LOAD
   //        DD DISP=SHR,DSN=qualif.HOME.LOAD
   //        DD DISP=SHR,DSN=DSN810.SDSNLOAD
   ```

   Preserve all other libraries. Submit the Rocket-generated JCL. Your COBOL2 program is linked to the Extender for Db2 interceptor modules. After the generated COBOL2 programs are compiled and link-edited, they can be run according to the documentation for Rocket QMF.

6. Allocate the Extender for Db2 dynamically via Call Attach during Rocket Compiler user interface invocation.

   Allocate the Extender for Db2 dynamically via Call Attach during Rocket Compiler user interface invocation. The Rocket Compiler for QMF generates QMF report programs in COBOL2. Users can run these generated programs from a panel, or they can invoke program execution by issuing a DSN RUN command from TSO. Both

methods require dynamic allocation of the ROCKET.QMF.LOAD library in STEPLIB, or ISPLLIB in the interactive TSO environment. Therefore, to enable the Rocket Compiler for QMF in a QMF/Db2 environment, users must set up a run-time allocation CLIST that allocates the ROCKET.QMF.LOAD library ahead in the concatenation sequence in STEPLIB, or ISPLLIB.

Also, the Rocket Compiler invokes the standard Db2 load library via a dynamic Call-Attach load. Therefore, to enable both the Rocket Compiler and the Extender for Db2 for QMF in a QMF/Db2 environment, you must set up STEPLIB, or ISPLLIB, of your run-time CLIST or JCL (log in proc) by placing the Extender for Db2 *qualif*.HOMEEXT.LOAD and *qualif*.HOME.LOAD load libraries before your Db2 load library. An example of a JCL STEPLIB allocation is:

```
//STEPLIB DD DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//        DD DISP=SHR,DSN=qualif.HOME.LOAD
//        DD DISP=SHR,DSN=qualif.ROCKET.QMF.LOAD
//        DD DISP=SHR,DSN=QMF810.SDSQLOAD
//        DD DISP=SHR,DSN=DSN810.SDSNEXIT
//        DD DISP=SHR,DSN=DSN810.SDSNLOAD
```

After these changes, standard Rocket users who dynamically allocate the Db2 standard load libraries via Call Attach dynamically allocate the Extender for Db2 *qualif*.HOMEEXT.LOAD and *qualif*.HOME.LOAD libraries ahead of the Db2 load library.

A CLIST version of the allocation can also be used. For an example of a run-time CLIST, see Installing the ibi WebFOCUS Extender for Db2 on z/OS. Make modifications as needed.

You can change the concatenation order of the load libraries in STEPLIB, as long as you follow these basic allocation conditions.

- For enabling the Extender for Db2, allocate the Extender for Db2 load libraries before the standard Db2 load library.
- For enabling the Rocket Compiler, allocate the Rocket load library before the standard QMF load library.

7. Verify the operation of the Extender for Db2 independent of the Rocket Compiler for QMF before enabling the two products.

Using COBOL

These examples illustrate how to set up a COBOL2 application that sends requests for data to a server through the Extender for Db2. The examples are:

- Sample COBOL2 Program
- Sample Link-Edit JCL

Sample COBOL2 Program

The sample COBOL2 program, XTDCOB, performs the following:

- Processes both PREPARED and EXECUTE IMMEDIATE statements.
- Accesses both local Db2 tables and server (Extender) tables.
- Accepts commands from SYSIN DD.
- Accepts parameters from XTDPRM DD.

XTDCOB can be run in the following modes:

- Batch JCL mode or interactive.
- DSNALI mode or DSNELI mode.

You can enter the following commands in SYSIN DD for a Prepared SQL statement or an EXECUTE IMMEDIATE statement.

```
+-----------------------------------------+
|            Available Commands:          |
|                                         |
| Prepared SQL statement Syntax:          |
|  SQL <sql statement>                    |
|  END                                    |
|                                         |
| Execute Immediate Syntax:               |
|  IMM <sql statement>                    |
|  END                                    |
|                                         |
| Type EXIT to exit this program.         |
+-----------------------------------------+
```

XTDCOB contains DEBUG and BATCH options for XTDPRM DD.

```
DEBUG=Y|N
```

where:

Y

Displays COBOL debugging messages.

N

Does not display the debugging messages. This is the default value.

```
BATCH=Y|N
```

where:

Y

Echoes input commands back to SYSOUT DD.

N

Does not echo input commands (interactive mode). This is the default value.

Note: The sample program can retrieve numeric and alpha columns, but it does not convert numeric columns into a displayable format. Therefore, the output of a numeric column appears in its internal binary representation. For example, the number 12 could be represented as 000C (undisplayable).

XTDCOB

A sample XTDCOB program is provided at **qualif**.HOME.DATA(XTDCOB) follows:

```
IDENTIFICATION DIVISION.
00370099
PROGRAM-ID. XTDCOB.
00380099
AUTHOR.     YANKO CASELLA.
00390099
ENVIRONMENT DIVISION.
00400099
```

```
CONFIGURATION SECTION.
00410099
SOURCE-COMPUTER. IBM-370.
00420099
OBJECT-COMPUTER. IBM-370.
00430099
INPUT-OUTPUT SECTION.
00440099
FILE-CONTROL.
00450099
    SELECT XTDPRM
00460099
          ASSIGN TO XTDPRM.
00470099
    SELECT DBGOUT
00471099
          ASSIGN TO DBGOUT.
00472099
DATA DIVISION.
00473099
FILE SECTION.
00474099
FD      XTDPRM
00475099
        RECORD CONTAINS 80 CHARACTERS
00476099
        BLOCK CONTAINS 0 RECORDS
00477099
        LABEL RECORDS ARE OMITTED
00478099
        RECORDING MODE IS F.
00479099
01  PRMREC                  PIC X(80).
00480099

00490099
FD      DBGOUT
00500099
        RECORD CONTAINS 132 CHARACTERS
00510099
        BLOCK CONTAINS 0 RECORDS
00520099
        LABEL RECORDS ARE OMITTED
00530099
        RECORDING MODE IS F.
00540099
01  MSGREC                  PIC X(132).
```

```
00550099

00550199
```

```
WORKING-STORAGE SECTION.
00570099
01  MSGERR                  PIC X(132).
00580099
77  MSGNUM                  PIC -ZZZ,ZZZ,ZZ9.
00590099
77  MSGBLK                  PIC X(132) VALUE SPACES.
00600099
77  MSGLEN                  PIC 9(4)   VALUE 132.
00610099

00610199
01 SLCT-STMT.
00610799
   49 SLCT-LENGTH           PIC S9(04) COMP-5.
00610899
   49 SLCT-STRING           PIC X(32700).
00610999

00611099
01 SYSIN-STRING.
00611499
   05 SYSIN-CMD             PIC X(04).
00611599
   05 SYSIN-LINE            PIC X(76).
00611699

00611799
```

```
01 REXX-PARMS.
00612099
   05 FILLER                PIC X(6) VALUE "DEBUG=".
00613099
   05 DEBUG-YES             PIC X.
00614099
   05 FILLER                PIC X(7) VALUE ",BATCH=".
00615099
   05 BATCH-YES             PIC X.
00616099
   05 FILLER                PIC X(65).
00617099
```

```
*******************************************************
00618099
* STRUCTURE FOR INPUT                                 *
00619099
*******************************************************
00620099
01 IOAREA.
00630099
   02 TNAME               PIC X(72).
00640099
   02 FILLER              PIC X(08).
00650099

00660099
*************************************************************
00670099
* VARIABLES FOR ERROR-MESSAGE FORMATTING                  *
00680099
*************************************************************
00690099
01 ERROR-MESSAGE.
00700099
   02 ERROR-LEN            PIC S9(4) COMP VALUE +960.
00710099
   02 ERROR-TEXT           PIC X(120) OCCURS 8 TIMES
00720099
                              INDEXED BY ERROR-INDEX.
00730099
77 ERROR-TEXT-LEN          PIC S9(8) COMP VALUE +120.
00740099
*************************************************************
00750099
*    SQLDA                                                *
00760099
*************************************************************
00770099
01 SQLDA.
00780099
   02 SQLDAID              PIC X(08)  VALUE "SQLDA   ".
00790099
   02 SQLDABC              PIC S9(08) COMP VALUE 33016.
00800099
   02 SQLN                 PIC S9(04) COMP VALUE 750.
00810099
   02 SQLD                 PIC S9(04) COMP VALUE 0.
00820099
   02 SQLVAR               OCCURS 1 TO 750 TIMES
```

```
00830099
                            DEPENDING ON SQLN.
00840099
     03 SQLTYPE             PIC S9(04) COMP.
00850099
     03 SQLLEN              PIC S9(04) COMP.
00860099
     03 SQLDATA             POINTER.
00870099
     03 SQLIND              POINTER.
00880099
     03 SQLNAME.
00890099
       49 SQLNAMEL          PIC S9(04) COMP.
00900099
       49 SQLNAMEC          PIC X(30).
00910099

00920099
77 VARCTYPE                 PIC S9(4) COMP VALUE +448.
00930099
77 CHARTYPE                 PIC S9(4) COMP VALUE +452.
00940099
77 VARLTYPE                 PIC S9(4) COMP VALUE +456.
00950099
77 VARGTYPE                 PIC S9(4) COMP VALUE +464.
00960099
77 GTYPE                    PIC S9(4) COMP VALUE +468.
00970099
77 LVARGTYP                 PIC S9(4) COMP VALUE +472.
00980099
77 FLOATYPE                 PIC S9(4) COMP VALUE +480.
00990099
77 DECTYPE                  PIC S9(4) COMP VALUE +484.
01000099
77 INTTYPE                  PIC S9(4) COMP VALUE +496.
01010099
77 HWTYPE                   PIC S9(4) COMP VALUE +500.
01020099
77 DATETYP                  PIC S9(4) COMP VALUE +384.
01030099
77 MDTTIMTP                 PIC S9(4) COMP VALUE +397.
01040099

01050099
```

```
01 TITLE-REC.
01060099
   02 TITLE-LEN             PIC S9(4) COMP.
01070099
   02 TITLE-LINE            PIC X(132).
01080099
   02 TITLE-SEP             PIC X(132) VALUE ALL "_".
01080199

01081099
01  SQLDATA-REC.
01091099
   02 REC1-LEN              PIC S9(8)  COMP.
01092099
   02 REC1-CHAR             PIC X(1) OCCURS 1 TO 32700 TIMES
01093299
        DEPENDING ON REC1-LEN.
01094099
01  SQLDATA-IND.
01100099
02 IND                      PIC S9(04) COMP OCCURS 750 TIMES.
01110099

01120099
01 RECPTR POINTER.
01130099
01 RECNUM REDEFINES RECPTR  PIC S9(9) COMP.
01140099
01 I                        PIC S9(4) COMP.
01150099
01 DUMMY                    PIC S9(4) COMP.
01160099
01 MYTYPE                   PIC S9(4) COMP.
01170099
01 COLUMN-IND               PIC S9(4) COMP.
01180099
01 COLUMN-LEN               PIC S9(4) COMP.
01190099
01 COLUMN-PREC              PIC S9(4) COMP.
01200099
01 COLUMN-SCALE             PIC S9(4) COMP.
01210099
01 INDCOUNT                 PIC S9(4) COMP.
01220099
01 ROWCOUNT                 PIC S9(9) COMP.
01230099
01 WORKAREA2.
```

```
01240099
   02 WORKINDPTR POINTER  OCCURS 750 TIMES.
01250099

01260099
  EXEC SQL
01270099
    DECLARE SLCT-CSR CURSOR FOR SLCT-CSR-STMT
01280099
  END-EXEC.
01290099

01300099
  EXEC SQL
01310099
     DECLARE SLCT-CSR-STMT STATEMENT
01320099
  END-EXEC.
01330099

01340099
  EXEC SQL INCLUDE SQLCA END-EXEC.
01350099

01360099
77  ONE                     PIC S9(4) COMP VALUE +1.
01380099
77  TWO                     PIC S9(4) COMP VALUE +2.
01390099
77  FOUR                    PIC S9(4) COMP VALUE +4.
01400099
77  QMARK                   PIC X          VALUE "?".
01401099
77  LAST-CMD                PIC X(4).
01410099

01420099
```

```
LINKAGE SECTION.
01430099
01  SQLDATA-BLANK.
01440099
   02 INDREC                PIC X(1).
01450099

01460099
```

```
PROCEDURE DIVISION.
01470099

01480099
****************************************************
01490099
* SQL RETURN CODE HANDLING                         *
01500099
****************************************************
01510099
  EXEC SQL WHENEVER SQLERROR  GOTO DBERROR END-EXEC.
01520099
  EXEC SQL WHENEVER NOT FOUND  CONTINUE    END-EXEC.
01540099

01550099
  OPEN INPUT XTDPRM.
01560099
  READ XTDPRM INTO REXX-PARMS.
01570099
  CLOSE XTDPRM.
01580099

01590099
  IF DEBUG-YES = 'Y' THEN
01600099
     OPEN OUTPUT DBGOUT.
01610099

01620099
  DISPLAY "+---------------------------------+".
01630099
  DISPLAY "|       Available Commands:       |".
01640099
  DISPLAY "|                                 |".
01650099
  DISPLAY "| Prepared SQL statement Syntax:  |".
01660099
  DISPLAY "|  SQL <sql statement>            |".
01670099
  DISPLAY "|  END                            |".
01680099
  DISPLAY "|                                 |".
01690099
  DISPLAY "| Execute Immediate Syntax:       |".
01700099
  DISPLAY "|  IMM <sql statement>            |".
```

```
01710099
  DISPLAY "|  END                            |".
01720099
  DISPLAY "|                                 |".
01730099
  DISPLAY "| Type EXIT to exit this program. |".
01740099
  DISPLAY "+---------------------------------+".
01750099
  DISPLAY " ".
01760099
  MOVE ONE TO SLCT-LENGTH.
01770099

01780099
MAIN-LOOP.
01790099

01800099
   DISPLAY "Type Command To Be Processed:".
01810099
   DISPLAY " ".
01820099
   MOVE SPACES TO LAST-CMD.
01830099

01840099
READ-SQL.
01850099
   ACCEPT  SYSIN-STRING FROM SYSIN.
01860099
   IF BATCH-YES = 'Y' THEN
01870099
     DISPLAY SYSIN-STRING.
01880099

01890099
```

```
EVALUATE FUNCTION UPPER-CASE (SYSIN-CMD)                      01900099
    WHEN "SQL "
01910099
      MOVE "SQL " TO LAST-CMD
01920099
      MOVE ONE TO SLCT-LENGTH
01930099
```

```
      STRING SYSIN-LINE DELIMITED BY SIZE
01940099
        INTO SLCT-STRING WITH POINTER SLCT-LENGTH
01950099
      GO TO READ-SQL
01960099

01970099
    WHEN "IMM "
01980099
       MOVE "IMM " TO LAST-CMD
01990099
       MOVE ONE TO SLCT-LENGTH
02000099
       STRING SYSIN-LINE DELIMITED BY SIZE
02010099
         INTO SLCT-STRING WITH POINTER SLCT-LENGTH
02020099
       GO TO READ-SQL
02030099

02040099
     WHEN "EXIT"
02050099
       GO TO PROG-END
02060099

02070099
     WHEN "END "
02080099
       SUBTRACT ONE FROM SLCT-LENGTH
02090099

02100099
       IF DEBUG-YES = 'Y' THEN
02110099
         STRING "LAST-CMD = ", LAST-CMD, MSGBLK
02120099
           DELIMITED BY MSGLEN INTO MSGERR
02130099
         WRITE MSGREC FROM MSGERR
02140099
         MOVE SLCT-LENGTH TO MSGNUM
02150099
         STRING "IN READ-SQL SLCT-LENGTH = ", MSGNUM, MSGBLK0
02160099
           DELIMITED BY MSGLEN INTO MSGERR
```

```
02170099
        WRITE MSGREC FROM MSGERR
02180099
        STRING "IN READ-SQL SLCT-STRING = ", SLCT-STRING,
02190099
           MSGBLK DELIMITED BY MSGLEN INTO MSGERR
02200099
        WRITE MSGREC FROM MSGERR
02210099
     END-IF
02220099

02230099
     EVALUATE LAST-CMD
02240099

02250099
       WHEN "SQL "
02260099
         PERFORM PROCESS-INPUT THROUGH IND-RESULT
02270099
       WHEN "IMM "
02280099
         PERFORM IMMED-SQL THROUGH PRINT-ROWS
02290099
     END-EVALUATE
02300099
     GO TO MAIN-LOOP
02310099

02320099
       WHEN OTHER
02330099
        STRING SYSIN-STRING DELIMITED BY SIZE
02340099
          INTO SLCT-STRING WITH POINTER SLCT-LENGTH
02350099
        GO TO READ-SQL
02360099
     END-EVALUATE.
02370099

02380099
```

```
     PROG-END.
02390099
```

```
       IF DEBUG-YES = 'Y' THEN
02400099
         CLOSE DBGOUT.
02410099
       GOBACK.
02420099
*********************************************************
02430099
* PREPARE                                               *
02440099
*********************************************************
02450099
PROCESS-INPUT.
02460099
  EXEC SQL
02470099
   PREPARE SLCT-CSR-STMT
02480099
    INTO :SQLDA
02490099
    FROM :SLCT-STMT
02500099
  END-EXEC.
02510099
*************************************************************
02520099
* SET UP ADDRESSES IN THE SQLDA FOR DATA                    *
02530099
*************************************************************
02540099
  IF DEBUG-YES = 'Y' THEN
02550099
   MOVE SQLD TO MSGNUM
02560099
   STRING "IN PROCESS-INPUT SQLD = ", MSGNUM, MSGBLK
02570099
   DELIMITED BY MSGLEN INTO MSGERR
02580099
   WRITE MSGREC FROM MSGERR
02590099
   MOVE SQLN TO MSGNUM
02600099
   STRING "IN PROCESS-INPUT SQLN = ", MSGNUM, MSGBLK
02610099
     DELIMITED BY MSGLEN INTO MSGERR
02620099
   WRITE MSGREC FROM MSGERR.
```

```
02630099
*************************************************************
02640099
* IF STATEMENT IS NOT SELECT, EXECUTE STMT                  *
02650099
*************************************************************
02660099
  IF SQLD = ZERO THEN
02670099
    GO TO NOT-A-SELECT.
02680099

02690099
  DISPLAY " "
02700099
  MOVE ZERO TO ROWCOUNT.
02710099
  MOVE ZERO TO REC1-LEN.
02720099
  SET  RECPTR TO ADDRESS OF REC1-CHAR(1).
02730099
  MOVE ONE TO I.
02740099
  INITIALIZE TITLE-LINE.
02740199
  MOVE ONE TO TITLE-LEN.
02741099
  PERFORM COLADDR UNTIL I > SQLD.
02750099
  MOVE SPACES TO SQLDATA-REC(5:REC1-LEN).
02750199
  DISPLAY TITLE-LINE.
02750299
  DISPLAY TITLE-SEP.
02751099
```

```
*********************************************************
02760099
*     SET LENGTH OF OUTPUT RECORD.                      *
02770099
 *     OPEN CURSOR                                      *
02780099
*********************************************************
02790099

02800099
```

```
   EXEC SQL OPEN SLCT-CSR END-EXEC.
02810099

02820099
*************************************************************
02830099
*                         FETCH                             *
02840099
*************************************************************
02850099
   IF DEBUG-YES = 'Y' THEN
02860099
   STRING "AT FETCH.....", MSGBLK DELIMITED BY MSGLEN
02870099
         INTO MSGERR
02880099
     WRITE MSGREC FROM MSGERR.
02890099

02900099
   EXEC SQL
02910099
     FETCH SLCT-CSR
02920099
     USING DESCRIPTOR :SQLDA
02930099
    END-EXEC.
02940099

02950099
   IF SQLCODE = ZERO THEN
02960099
    PERFORM WRITE-AND-FETCH
02970099
      UNTIL SQLCODE IS NOT EQUAL TO ZERO.
02980099

02990099
   MOVE ROWCOUNT TO MSGNUM
03000099
   DISPLAY " "
03010099
   DISPLAY "***** NUMBER OF RECORDS IN TABLE=" MSGNUM " *****"
03020099
   DISPLAY " "
03030099
```

```
03040099
   IF DEBUG-YES = 'Y' THEN
03050099
    STRING "LEAVING FETCH.....", MSGBLK
03060099
        DELIMITED BY MSGLEN INTO MSGERR
03070099
    WRITE MSGREC FROM MSGERR.
03080099

03090099
CLOSEDT.
03100099
   IF DEBUG-YES = 'Y' THEN
03110099
    STRING "AT CLOSEDT.....", MSGBLK
03120099
       DELIMITED BY MSGLEN INTO MSGERR
03130099
    WRITE MSGREC FROM MSGERR.
03140099

03150099
   EXEC SQL CLOSE SLCT-CSR END-EXEC.
03160099

03170099
IND-RESULT.
03180099
   IF DEBUG-YES = 'Y' THEN
03190099
    STRING "AT IND-RESULT. RETURNING TO MAIN LOOP.", MSGBLK
03200099
       DELIMITED BY MSGLEN INTO MSGERR
03210099
    WRITE MSGREC FROM MSGERR.
03220099

03230099
```

```
WRITE-AND-FETCH.
03240099
   IF DEBUG-YES = 'Y' THEN
03250099
    STRING "AT WRITE-AND-FETCH.....", MSGBLK
03260099
```

```
       DELIMITED BY MSGLEN INTO MSGERR
03270099
    WRITE MSGREC FROM MSGERR.
03280099

03290099
   MOVE ONE TO INDCOUNT.
03300099
   PERFORM NULLCHK UNTIL INDCOUNT > SQLD.
03310099

03320099
   IF DEBUG-YES = 'Y' THEN
03330099
     STRING "SQLDATA-REC = ", SQLDATA-REC, MSGBLK
03340099
        DELIMITED BY MSGLEN INTO MSGERR
03350099
     WRITE MSGREC FROM MSGERR.
03360099

03370099
   DISPLAY SQLDATA-REC(5:REC1-LEN).
03380099
   MOVE SPACES TO SQLDATA-REC(5:REC1-LEN).
03390099
   ADD ONE TO ROWCOUNT.
03420099

03430099
   EXEC SQL
03440099
     FETCH SLCT-CSR
03450099
        USING DESCRIPTOR :SQLDA
03460099
     END-EXEC.
03470099

03480099
   IF DEBUG-YES = 'Y' THEN
03490099
     STRING "IN WRITE-AND-FETCH SQLDA = ", SQLDA, MSGBLK
03500099
        DELIMITED BY MSGLEN INTO MSGERR
03510099
       WRITE MSGREC FROM MSGERR
```

```
03520099

03530099
       STRING "LEAVING WRITE-AND-FETCH.....", MSGBLK
03540099
        DELIMITED BY MSGLEN INTO MSGERR
03550099
       ITE MSGREC FROM MSGERR.
03560099
NULLCHK.
03570099
   IF DEBUG-YES = 'Y' THEN
03580099
     STRING "AT NULLCHK.....", MSGBLK
03590099
        DELIMITED BY MSGLEN INTO MSGERR
03600099
     WRITE MSGREC FROM MSGERR
03610099

03620099
     MOVE IND(INDCOUNT) TO MSGNUM
03630099
     STRING "IN NULLCHK IND(INDCOUNT) = ", MSGNUM, MSGBLK
03640099
        DELIMITED BY MSGLEN INTO MSGERR
03650099
     WRITE MSGREC FROM MSGERR.
03660099

03690199
   IF IND(INDCOUNT) < 0 THEN
03691099
     SET ADDRESS OF SQLDATA-BLANK TO WORKINDPTR(INDCOUNT)
03692099
     MOVE QMARK TO INDREC.
03700099

03710099
```

```
   IF DEBUG-YES = 'Y' THEN
03720099
```

```
     MOVE INDCOUNT TO MSGNUM
03730099
```

```
     STRING "IN NULLCHK AFTER IF - INDCOUNT = ", MSGNUM, MSGBLK
03740099
         DELIMITED BY MSGLEN INTO MSGERR
03750099
     WRITE MSGREC FROM MSGERR.
03760099

03770099
   ADD ONE TO INDCOUNT.
03780099

03790099
   IF DEBUG-YES = 'Y' THEN
03800099
       STRING "LEAVING NULLCHK....", MSGBLK
03810099
          DELIMITED BY MSGLEN INTO MSGERR
03820099
       WRITE MSGREC FROM MSGERR.
03830099

03850699
COLADDR.
03851099
   IF DEBUG-YES = 'Y' THEN
03860099
       STRING "AT COLADDR.....", MSGBLK
03870099
          DELIMITED BY MSGLEN INTO MSGERR
03880099
       WRITE MSGREC FROM MSGERR.
03890099

03900099
   SET SQLDATA(I) TO RECPTR.
03940099
*****************************************************************
03950099
*             DETERMINE LENGTH OF COLUMN (COLUMN-LEN)          *
03960099
*****************************************************************
03970099
   MOVE SQLLEN(I) TO COLUMN-LEN.
03980099
*****************************************************************
03990099
*             COLUMN-IND IS 0 FOR NO NULLS AND 1 FOR NULLS     *
```

```
04000099
*************************************************************
04010099
    DIVIDE SQLTYPE(I) BY TWO GIVING DUMMY REMAINDER COLUMN-IND.
04020099
*************************************************************
04030099
*             MYTYPE IS JUST THE SQLTYPE WITHOUT THE NULL BIT  *
04040099
*************************************************************
04050099
    MOVE SQLTYPE(I) TO MYTYPE.
04060099

04070099
    IF DEBUG-YES = 'Y' THEN
04080099
       MOVE SQLTYPE(I) TO MSGNUM
04090099
       STRING "IN COLADDR SQLTYPE(I) = ", MSGNUM, MSGBLK
04100099
           DELIMITED BY MSGLEN INTO MSGERR
04110099
       WRITE MSGREC FROM MSGERR.
04120099

04130099
    SUBTRACT COLUMN-IND FROM MYTYPE.
04140099
```

```
*************************************************************
04150099
*          SET THE COLUMN LENGTH, DEPENDENT UPON DATA TYPE *
04160099
*************************************************************
04170099
    EVALUATE MYTYPE
04180099
       WHEN CHARTYPE CONTINUE,
04190099
       WHEN DATETYP THROUGH MDTTIMTP CONTINUE,
04200099
       WHEN FLOATYPE CONTINUE,
04210099
       WHEN VARCTYPE
04220099
```

```
          ADD TWO TO COLUMN-LEN,
04230099
       WHEN VARLTYPE
04240099
          ADD TWO TO COLUMN-LEN,
04250099
       WHEN GTYPE
04260099
          MULTIPLY COLUMN-LEN BY TWO GIVING COLUMN-LEN,
04270099
       WHEN VARGTYPE
04280099
          PERFORM CALC-VARG-LEN,
04290099
       WHEN LVARGTYP
04300099
          PERFORM CALC-VARG-LEN,
04310099
       WHEN HWTYPE
04320099
          MOVE TWO TO COLUMN-LEN,
04330099
       WHEN INTTYPE
04340099
          MOVE FOUR TO COLUMN-LEN,
04350099
       WHEN DECTYPE
04360099
          PERFORM CALC-DECIMAL-LEN,
04370099
       WHEN OTHER
04380099
          PERFORM UNRECOGNIZED-ERROR,
04390099
    END-EVALUATE.
04400099

04410099
    IF DEBUG-YES = 'Y' THEN
04420099
       MOVE COLUMN-LEN TO MSGNUM
04430099
       STRING "IN COLADDR COLUMN-LEN = ", MSGNUM, MSGBLK
04440099
          DELIMITED BY MSGLEN INTO MSGERR
04450099
       WRITE MSGREC FROM MSGERR.
```

```
04460099

04470099
    ADD COLUMN-LEN TO RECNUM.
04480099
    ADD COLUMN-LEN TO REC1-LEN.
04490099

04490199
    STRING SQLNAMEC(I) DELIMITED BY SPACE
04491099
           INTO TITLE-LINE POINTER TITLE-LEN.
04491199
    STRING SPACE DELIMITED BY SIZE
04491299
           INTO TITLE-LINE POINTER TITLE-LEN.
04491399
```

```
************************************************************
04500099
*IF THIS COLUMN CAN BE NULL, AN INDICATOR VARIABLE IS NEEDED*
04510099
************************************************************
04520099
    MOVE ZERO TO IND(I)
04530099
    IF COLUMN-IND = ONE THEN
04540099
       SET SQLIND(I) TO ADDRESS OF IND(I)
04550099
       SET WORKINDPTR(I) TO RECPTR
04560099
       ADD ONE TO RECNUM
04570099
       ADD ONE TO REC1-LEN.
04580099

04583099
    ADD ONE TO I.
04590099

04600099
    IF DEBUG-YES = 'Y' THEN
04610099
       STRING "LEAVING COLADDR....", MSGBLK
04620099
```

```
              DELIMITED BY MSGLEN INTO MSGERR
04630099
       WRITE MSGREC FROM MSGERR.
04640099
*************************************************************
04650099
*CALCULATE COLUMN LENGTH FOR A DECIMAL DATA TYPE COLUMN.    *
04660099
*************************************************************
04670099
CALC-DECIMAL-LEN.
04680099
    IF DEBUG-YES = 'Y' THEN
04690099
       STRING "AT CALC-DECIMAL-LEN....", MSGBLK
04700099
          DELIMITED BY MSGLEN INTO MSGERR
04710099
       WRITE MSGREC FROM MSGERR.
04720099

04730099
    DIVIDE COLUMN-LEN BY 256 GIVING COLUMN-PREC
04740099
                      REMAINDER COLUMN-SCALE.
04750099
    MOVE COLUMN-PREC TO COLUMN-LEN.
04760099
    ADD ONE TO COLUMN-LEN.
04770099
    DIVIDE COLUMN-LEN BY TWO GIVING COLUMN-LEN.
04780099

04790099
    IF DEBUG-YES = 'Y' THEN
04800099
       MOVE COLUMN-LEN TO MSGNUM
04810099
       STRING "IN CALC-DECIMAL-LEN COLUMN-LEN= ", MSGNUM, MSGBLK
04820099
          DELIMITED BY MSGLEN INTO MSGERR
04830099
       WRITE MSGREC FROM MSGERR
04840099

04850099
       STRING "LEAVING CALC-DECIMAL-LEN....", MSGBLK
```

```
04860099
              DELIMITED BY MSGLEN INTO MSGERR
04870099
       WRITE MSGREC FROM MSGERR.
04880099
```

```
************************************************************
04890099
*PERFORM PARAGRAPH TO CALCULATE COLUMN LENGTH              *
04900099
*FOR A VARGRAPHIC DATA TYPE COLUMN.                        *
04910099
************************************************************
04920099
CALC-VARG-LEN.
04930099
    IF DEBUG-YES = 'Y' THEN
04940099
       STRING "AT CALC-VARG-LEN.....", MSGBLK
04950099
          DELIMITED BY MSGLEN INTO MSGERR
04960099
       WRITE MSGREC FROM MSGERR.
04970099

04980099
    MULTIPLY COLUMN-LEN BY TWO GIVING COLUMN-LEN.
04990099
    ADD TWO TO COLUMN-LEN.
05000099

05010099
    IF DEBUG-YES = 'Y' THEN
05020099
       STRING "LEAVING CALC-VARG-LEN.....", MSGBLK
05030099
          DELIMITED BY MSGLEN INTO MSGERR
05040099
       WRITE MSGREC FROM MSGERR.
05050099
************************************************************
05060099
*PERFORM PARAGRAPH TO NOTE AN UNRECOGNIZED DATA TYPE COLUMN.*
05070099
************************************************************
05080099
```

```
UNRECOGNIZED-ERROR.
05090099
    MOVE MYTYPE TO MSGNUM
05100099
    DISPLAY "UNRECOGNIZED DATA TYPE = " MSGNUM
05110099
    MOVE COLUMN-LEN TO MSGNUM
05120099
    DISPLAY "            COLUMN-LEN = " MSGNUM
05130099

05140099
    GO TO IND-RESULT.
05150099

05160099
NOT-A-SELECT.
05170099
    IF DEBUG-YES = 'Y' THEN
05180099
       STRING "AT NOT-A-SELECT....", MSGBLK
05190099
          DELIMITED BY MSGLEN INTO MSGERR
05200099
       WRITE MSGREC FROM MSGERR.
05210099

05220099
    EXEC SQL
05230099
       EXECUTE SLCT-CSR-STMT USING DESCRIPTOR :SQLDA
05240099
    END-EXEC.
05250099

05260099
    PERFORM PRINT-ROWS.
05270099

05280099
    GO TO IND-RESULT.
05290099

05300099
```

```
IMMED-SQL.
05310099
    IF DEBUG-YES = 'Y' THEN
05320099
       STRING "AT IMMED-SQL....", MSGBLK
05330099
          DELIMITED BY MSGLEN INTO MSGERR
05340099
       WRITE MSGREC FROM MSGERR.
05350099

05360099
   EXEC SQL
05370099
      EXECUTE IMMEDIATE :SLCT-STMT
05380099
   END-EXEC.
05390099

05400099
PRINT-ROWS.
05410099
    MOVE SQLERRD(3) TO MSGNUM
05420099
    DISPLAY " "
05430099
    DISPLAY "***** NUMBER OF ROWS AFFECTED BY REQUEST=" MSGNUM
05440099
                                                        " *****"
05450099
    DISPLAY " "
05451099
    IF DEBUG-YES = 'Y' THEN
05451199
       STRING "LEAVING PRINT-ROWS....", MSGBLK
05451299
          DELIMITED BY MSGLEN INTO MSGERR
05451399
       WRITE MSGREC FROM MSGERR.
05451499

05451599
DBERROR.
05451699
    IF DEBUG-YES = 'Y' THEN
05451799
       STRING "AT DBERROR.....", MSGBLK
```

```
05451899
          DELIMITED BY MSGLEN INTO MSGERR
05451999
       WRITE MSGREC FROM MSGERR
05452099
       MOVE SQLCODE TO MSGNUM
05456099
       STRING "SQL ERROR OCCURRED, SQLCODE = ", MSGNUM,
05457099
          MSGBLK DELIMITED BY MSGLEN INTO MSGERR
05458099
       WRITE MSGREC FROM MSGERR.
05459099

05460099
    CALL "DSNTIAR" USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN.
05470099

05480099
    IF RETURN-CODE = ZERO
05490099
       PERFORM ERROR-PRINT VARYING ERROR-INDEX
05500099
          FROM 1 BY 1 UNTIL ERROR-INDEX GREATER THAN 8
05510099
```

```
    ELSE
05520099
       IF DEBUG-YES = 'Y' THEN
05530099
          STRING "DSNT497I RETURN CODE FROM MSG ROUTINE
DSNTIAR",05540099
             MSGBLK DELIMITED BY MSGLEN INTO MSGERR
05550099
       WRITE MSGREC FROM MSGERR
05560099
     END-IF
05570099
    END-IF.
05580099

05590099
    IF DEBUG-YES = 'Y' THEN
05600099
       STRING "LEAVING DBERROR....", MSGBLK
```

```
05610099
             DELIMITED BY MSGLEN INTO MSGERR
05620099
       WRITE MSGREC FROM MSGERR.
05630099

05640099
    IF LAST-CMD = "SQL " THEN
05650099
       GO TO IND-RESULT
05660099
    ELSE
05661099
       GO TO PRINT-ROWS.
05662099

05663099
ERROR-PRINT.
05664099
    IF DEBUG-YES = 'Y' THEN
05665099
       STRING "AT ERROR-PRINT....", MSGBLK
05666099
             DELIMITED BY MSGLEN INTO MSGERR
05667099
       WRITE MSGREC FROM MSGERR.
05668099

05669099
    DISPLAY ERROR-TEXT (ERROR-INDEX).
05670099

05680099
    IF DEBUG-YES = 'Y' THEN
05690099
      STRING "LEAVING ERROR-PRINT....", MSGBLK
05700099
             DELIMITED BY MSGLEN INTO MSGERR
05710099
      WRITE MSGREC FROM MSGERR.
05720099
```

Sample Link-Edit JCL

This JCL sample, XTDCLG, precompiles, compiles, link-edits, and runs COBOL2 programs containing dynamic SQL with the Extender for Db2. XTDCLG does the following.

- Precompiles, compiles, and links the program XTDCOB. Program XDTCOB accepts commands from SYSIN DD, either in batch or interactively (CLIST). It can be run in either DSNALI or DSNELI mode.

  The syntax for SYSIN DD is:

```
+-----------------------------------------+
|            Available Commands:          |
|                                         |
| Prepared SQL statement Syntax:          |
|  SQL <sql statement>                    |
|  END                                    |
|                                         |
| Execute Immediate Syntax:               |
|  IMM <sql statement>                    |
|  END                                    |
|                                         |
| Type EXIT to exit this program.         |
+-----------------------------------------+
```

- Runs XTDCOB.

To link-edit your JCL, perform the following steps.

1. Change all instances of **qualif**, **user**, **db2hlq,dbss**, **hostn**, and **portn** to match your site specifications. See comments in the jcl at **qualif**.HOME.DATA(XTDCLG).
2. Copy COBOL source member XTDCOB from qualif.HOME.DATA into your COBOL source library. Make the necessary changes to the SYSIN DD card in the PC step.
3. Confirm that you have followed the installation instructions for the Extender for Db2 as described in Chapter 2, *Installing the Extender for Db2 on z/OS*. Verify that you have link-edited the Extender for Db2 main module with your Db2 entry points, if your site has a local Db2 subsystem. If you have properly link-edited the Extender for Db2 main module with your Db2 entry points, the **qualif**.HOMEEXT.LOAD library referenced in the STEPLIB of the RUNSTEP in the XTDCLG JCL should be properly linked to your site's local Db2 subsystem.
4. Submit the JCL. The output of the SQL request is found in SYSOUT.

Note: If an application only processes servers (no local Db2 access), it is not necessary to bind and grant the customer application plan. The server default plan (dynamic plan) is used instead.

XTDCLG

```
//*        Job Card Goes Here
//*
//* Note: DSNELI could be used instead, DSNALI was used      *
//*       arbitrarily.                                       *
//*                                                          *
//*Substitutions:                                            *
//*  qualif - High level qualifier for DB2 Extender datasets *
//*  db2hlq - High level qualifier for DB2 libraries.        *
//*  user   - High level qualifier for user libraries.       *
//*  dbss   - DB2 Subsytem name.                             *
//*  hostn  - Server's Host name or Server's IP address.     *
//*  portn  - TCP/IP Port number server is listening on.     *
//************************************************************
//         SET DB2REL=db2hlq
//************************************************************
//*                 PC XDTCOB
//************************************************************
//PC       EXEC  PGM=DSNHPC,
//         PARM='HOST(COB2),QUOTE,APOSTSQL,ATTACH(CAF)'
//STEPLIB  DD  DISP=SHR,DSN=&DB2REL..SDSNEXIT
//         DD  DISP=SHR,DSN=&DB2REL..SDSNLOAD
//SYSIN    DD  DISP=SHR,DSN=user.COBOL.SOURCE(XTDCOB)
//DBRMLIB  DD  DISP=SHR,DSN=user.DBRMLIB.DATA(XTDCOB)
//SYSCIN   DD  DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=SYSDA,
//         SPACE=(800,(500,500))
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT2   DD  UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT3   DD  UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT4   DD  UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
//SYSUT5   DD  UNIT=SYSDA,SPACE=(800,(500,500),,,ROUND)
```

```
//*********************************************************
//*           C O M P I L E
//*********************************************************
```

```
//COB      EXEC  PGM=IGYCRCTL,COND=(4,LT),
//         PARM='QUOTE,OBJECT,MAP,LIST,RENT,NODYNAM'
//SYSIN    DD  DSN=&&DSNHOUT,DISP=(OLD,DELETE)
//SYSLIN   DD  DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//         SPACE=(800,(500,500))
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT6   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT7   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//*******************************************************
//*               LINKEDIT STEP FOR XTDCOB
//*******************************************************
//LKEDXTD  EXEC  PGM=IEWL,PARM='XREF',COND=(4,LT)
//SYSLIB   DD  DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//         DD  DISP=SHR,DSN=&DB2REL..SDSNLOAD
//         DD  DISP=SHR,DSN=CEE.SCEELKED
//SYSLMOD  DD  DISP=SHR,DSN=user.COBOL.LOAD
//OBJECT   DD  DSN=&&LOADSET,DISP=(OLD,DELETE,DELETE)
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSLIN   DD  *
  INCLUDE  SYSLIB(DSNALI)
  INCLUDE  SYSLIB(DSNTIAR)
  INCLUDE  OBJECT
  MODE     AMODE(31),RMODE(ANY)
  ENTRY    XTDCOB
  NAME     XTDCOB(R)
/*
```

```
//**************************************************
//*               BIND STEP FOR XTDCOB
//* (only required to use DB2 directly (NATIVELY))
//**************************************************
//BIND      EXEC   PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB   DD DSN=&DB2REL..SDSNLOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//DBRMLIB  DD  DISP=SHR,DSN=user.DBRMLIB.DATA(XTDCOB)
//SYSTSIN   DD *
DSN SYSTEM(dbss)
BIND PLAN      (XTDCOB)   -
     MEMBER    (XTDCOB)   -
```

```
     LIBRARY   ('user.DBRMLIB.DATA') -
     ACTION    (REPLACE) -
     ISOLATION (CS) -
     ACQUIRE   (USE) -
     RELEASE   (COMMIT) -
     EXPLAIN   (YES)
END
/*
//*****************************************************
//*               GRANT STEP FOR XTDCOB
//* (only required to use DB2 directly (NATIVELY))
//*****************************************************
//GRANT     EXEC   PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB   DD DSN=&DB2REL..SDSNLOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
RUN PROGRAM (DSNTIAD) PLAN (DSNTIA81) -
    LIB ('DSN810.RUNLIB.LOAD')
END
/*
//SYSIN     DD *
          GRANT EXECUTE ON PLAN XTDCOB TO PUBLIC;
/*
```

```
//*******************************************************************
//*             RUN STEP for XTDCOB
//*******************************************************************
//RUNXTD   EXEC PGM=XTDCOB,COND=((4,LT),EVEN)
//STEPLIB   DD  DISP=SHR,DSN=user.COBOL.LOAD
//          DD  DISP=SHR,DSN=qualif.HOMEEXT.LOAD
//          DD  DISP=SHR,DSN=qualif.HOME.LOAD
//          DD  DISP=SHR,DSN=&DB2REL..SDSNEXIT
//          DD  DISP=SHR,DSN=&DB2REL..SDSNLOAD
//EDACS3    DD  *
NAME        = Client Odin File
NODE = EDASERVE
  BEGIN
    PROTOCOL  = TCP
    CLASS     = CLIENT
    HOST      = hostn    ;Server's Host name or IP address
    PORT      = portn    ;Port # server is listening on
;   TRACE     = 31
  END
/*
//EDAENV    DD  *
```

```
FSTRACE=DD:FSTRACE
EDACONF=/PDS
/*
//EDADPDS   DD  DUMMY
//*EDAPARMS DD  DISP=SHR,DSN=user.EDAPARMS
//IBITRACE  DD  *
SET TRACEON=ALL
/*
//FSTRACE   DD  SYSOUT=*,DCB=(LRECL=132,RECFM=FB,BLKSIZE=132)
//SYSOUT    DD  SYSOUT=*
//DBGOUT    DD  SYSOUT=*
//XTDPRM    DD  *
DEBUG=N,BATCH=Y
/*
//SYSIN     DD  *
SQL
SELECT COUNTRY,CAR,MODEL,BODYTYPE FROM EDASERVE.ANYNAME.CAR
END
SQL
SELECT LAST_NAME,FIRST_NAME FROM EDASERVE.ANYNAME.EMPLOYEE
END
EXIT
/*
```

ibi WebFOCUS Extender for Db2 Error Messages and Codes

This section lists the server and Extender for Db2 messages and codes. It also provides a cross-reference to Db2 SQLCODES.

API Status Codes

API status codes are converted to SQLCODEs and SQLERRMC (which reside in SQLCA). The following chart shows some of the API status codes and conversions.

API Status Code	SQLCODE
0	0
5	+100
-9	-904 for Db2
-12	-904 for Db2
Most other negative server status codes	Default ERRNUM (if not defined, then -901)

ibi WebFOCUS Reporting Server Error Codes and SQLCODEs

The following table illustrates some of the current conversions of server codes to SQLCODEs.

Server Code	SQLCODE	Meaning
EDA0000	0	Status OK
EDA0003	-206	Field not found
EDA0005	-840	Too many columns
EDA0010	-136	Too many sort keys
EDA0016	-206	Field not found
EDA0201	-802	Division by zero
EDA0202	-802	Floating point overflow
EDA0203	0	Floating point underflow
EDA0205	-204	File not found
EDA0236	-206	Field not found
EDA0258	-206	Field not found
EDA0277	-414	Invalid format in LIKE column
EDA0281	-401	Incompatible types
EDA0370	-206	Field not found
EDA0486	0	File allocated
EDA0582	0	SQL syntax error
EDA0757	0	Multi-path DB-ignore
EDA4204	-204	No logical PCB for file
EDA4211	-204	No BMP region has PCB

Server Code	SQLCODE	Meaning
EDA4902	0	Model 204 database restarted; informational
EDA14007	-104	Syntax error
EDA14009	-204	File not found
EDA14010	-206	Field not found
EDA14012	-203	Column name ambiguous
EDA14013	-84	Unsupported SQL syntax
EDA14014	-401	Incompatible operands
EDA14015	-412	Too many columns on subquery
EDA14018	-132	Invalid pattern on LIKE
EDA14025	-421	Incompatible UNION
EDA14026	-415	Incompatible UNION
EDA14028	-125	Invalid ORDER BY number
EDA14030	-601	Table exists on CREATE
EDA14041	-313	Too many parameter markers
EDA14043	-612	Duplicate column name
EDA14045	-601	View exists on CREATE
EDA14046	-204	View not found
EDA14053	-122	Invalid GROUP BY clause

Server Code	SQLCODE	Meaning
EDA14056	-122	Invalid GROUP BY clause
EDA14058	-127	Invalid use of DISTINCT
EDA14064	-112	Invalid aggregate function
EDA14065	-120	Invalid aggregate function
EDA14066	-129	Too many tables in SQL

ibi WebFOCUS Extender for Db2 Error Codes

The Extender can generate its own error codes into a specific SQLCODE and SQLERRMC (in SQLCA). The following are some of the Extender-generated Db2 codes.

Server Event	Db2 Code
An overflow is detected.	+802
An unacceptable SQL statement is found.	-084
Too many tables or views are in a request.	-129
A null value cannot be assigned to output host variable because no indicator variable is specified.	-305
The number of parameter markers does not match the number of SQLDA entries.	-313
Unsupported use of parameter markers.	-418
A cursor in a FETCH or CLOSE statement is not OPEN.	-501

Server Event	Db2 Code
A cursor in an OPEN statement is already OPEN.	-502
An SQL request references multiple locations.	-512
A DESCRIBE was performed on an UNPREPAREd statement.	-516
A PREPARE statement identifies the SELECT statement of the OPENed cursor.	-519
A user ID does not have privilege to perform operation or the server is Read/Only.	-551
An attempt is made to CONNECT when an application is not in a CONNECTable state.	-752
An overflow is detected.	-802
The main module QXQMFX has not been linked to IBM DSNALI, DSNHLI, DSNTIAR.	-901
With Db2, an unsuccessful execution is caused by an unavailable resource. Most likely, the server is not connected.	-904
With SQL/DS, an unsuccessful execution is caused by an unavailable resource. Most likely, the server is not connected.	-940

Connecting to Multiple ibi WebFOCUS Reporting Servers

The Extender for Db2 supports the use of SQL CONNECT to connect to multiple servers or to retrieve information about the currently connected server. Use the command to connect to a server that you specify or to the server specified in the EDAPARMS file.

The Db2 connection states of connectable and connected/unconnected, and of unconnectable and connected are similar to the Db2 design and fully supported. For more information, see the *IBM Db2 Reference Manual*. Use SQL CONNECT functionality in the same manner as in a multiple Db2 subsystem environment.

Explicitly Connecting to a ibi WebFOCUS Reporting Server

Use the following form of SQL CONNECT to establish the *current server* for the client application process. The current server must be a valid server or a valid Db2 subsystem.

```
EXEC SQL CONNECT TO :host variable
EXEC SQL CONNECT TO location
```

where:

host variable

Is a host variable.

location

Is an explicit location. For the explicit designation of the current server, the location name must be a valid server or a valid Db2 subsystem.

- To be a valid server, its definition must reside under the communications

configuration file.

- For a valid Db2 subsystem, the Extender for Db2 does not recognize any local or remote Db2 subsystems, but passes the CONNECT request to Db2, for Db2 to make the validation. In this manner, remote Db2 subsystems can be connected to the Extender for Db2, provided that the local Db2 subsystem, which the Extender for Db2 was originally link-edited at installation can recognize the remote Db2 subsystem name.

On a COMMIT or ROLLBACK, the Extender for Db2 resolves and dispatches partially-qualified table requests to the *current server*.

Use the explicit form of SQL CONNECT whenever partially-qualified tables must be resolved to a Server for Db2 or to a server.

Implicitly Connecting to a ibi WebFOCUS Reporting Server

Use the following form of SQL CONNECT to establish the server defined in the EDAPARMS file as the *current server*:

```
EXEC SQL CONNECT RESET
```

Issuing this command is equivalent to issuing.

```
EXEC SQL CONNECT TO default server
```

where:

default server

Is the server defined in the EDAPARMS file.

On a COMMIT or ROLLBACK, the Extender for Db2 resolves and dispatches partially-qualified table requests to the current server.

The default server is defined in the EDAPARMS file under the EDASERVE keyword. If the local Db2 subsystem is the desired default, then omit the EDASERVE assignment in the EDAPARMS file.

Retrieving Information About a ibi WebFOCUS Reporting Server

Use the following form of SQL CONNECT to return information about the current server in the SQLERRP field of the SQLCA.

```
EXEC SQL CONNECT
```

For the Extender for Db2, the information returned in the SQLERRP field is:

```
AKB02000
```

The connected server or location may be explicitly or implicitly defined.

Error Messages

An unsuccessful SQL CONNECT TO request returns a SQLCODE of -752, and an SQLSTATE of 51011 in the SQLCA.

ibi Documentation and Support Services

For information about this product, you can read the documentation, contact Support, and join Community.

How to Access ibi Documentation

Documentation for ibi products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The documentation for this product is available on the ibi™ WebFOCUS® Reporting Server Documentation page.

How to Contact Support for ibi Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join ibi Community

ibi Community is the official channel for ibi customers, partners, and employee subject matter experts to share and access their collective experience. ibi Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from ibi products. For a free registration, go to ibi Community.

Legal and Third-Party Notices

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.