NPL GATEWAY TO ODBC

PROGRAMMER'S GUIDE

1st Edition - July 1998

Niakwa, Inc.
23600 N. Milwaukee Avenue
Vernon Hills, IL 60061

PHONE (847) 634-8700 FAX (847) 634-8718

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES AND PROPRIETARY RIGHTS

The staff of Niakwa, Inc. (Niakwa) has taken due care in preparing this manual. Nothing contained herein shall be construed to modify or alter in any way the standard terms and conditions of the Niakwa Programming Language (NPL) Support and Distribution License Agreement, the End-User Support Only License Agreement, the Niakwa Software License Agreement and Warranty and any other Niakwa License Agreement (collectively, the "License Agreements") by which this software package was acquired.

This manual is to serve as a guide for use of the Niakwa software only and not as a source of representations or additional undertakings by Niakwa. The licensee must refer to the License Agreements for Niakwa product and service representations.

No ownership of Niakwa software is transferred by any of the License Agreements. Any use of Niakwa software beyond the terms and conditions of the License Agreements, without the written authorization of Niakwa, is prohibited.

PREFACE

P.1 PREREQUISITE KNOWLEDGE

The following documentation assumes a thorough understanding of the NPL MS-Windows Release V software and documentation.

While it is not necessary to learn or comprehend the software language C, a conceptual understanding of Microsoft's ODBC 3.0 and SQL (Structured Query Language) is required.

To use the NPL Gateway to ODBC Revision 1.20, a developer will require the following:

NPL for MS-Windows 32-bit RunTime (Revision 5.00.05 or greater).

NPL for MS-DOS based Operating Environments Development Package.

An installed ODBC driver for the database to be supported. These are commercially available through several sources like Q+E.

Microsoft's ODBC 3.0

NOTE: The Microsoft ODBC 3.0 SDK is required for its documentation and the drivers required to allow testing of applications for driver inconsistencies. This manual focuses only on Niakwa's implementation of its ODBC library and defines the ODBC functions implemented.

A basic understanding of SQL commands and operation.

NOTE: The NPL Gateway to ODBC does not does not require the Niakwa Gateway to MS-Windows , although they do share a common external library of functions.

P.2 HOW TO USE THIS MANUAL

This manual should be used by developers as a guide on how to create and modify NPL applications using the NPL Gateway to ODBC product.

All chapters should be reviewed thoroughly by the developer. The following is a summary of the topics discussed in each chapter.

Chapter 1 is an introduction to the NPL Gateway to ODBC and discusses specific features of the product.

Chapter 2 discussed the installation and configuration of the NPL Gateway to ODBC.

Chapter 3 discusses developing routines in NPL using the NPL Gateway to ODBC.

Chapter 4 discusses the ODBC function calls.

Chapter 5 discusses the Niakwa provided NPL Gateway to ODBC examples.

Chapter 6 discusses end-user considerations.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

The 32-bit NPL Gateway to ODBC is a Niakwa Development Tool product that allows developers to make calls to any MS-Windows based 32-bit ODBC database driver. The NPL Gateway to ODBC is fully compliant with Microsoft's ODBC 3.0. Calling conventions (function names and parameter lists) of the Niakwa ODBC are nearly 100% compatible with the native C ODBC API.

In addition, this product includes a generic interface to third party .DLL files that allows NPL developers to access third-party ODBC drivers without writing code in C.

Section 1.2 discusses the contents of the NPL Gateway to ODBC.

Section 1.3 discusses the benefits of the NPL Gateway to ODBC.

Section 1.4 discusses developer considerations for using the NPL Gateway to ODBC.

Section 1.5 discusses the components of the NPL Gateway to ODBC.

1.2 CONTENTS OF THE NPL GATEWAY TO ODBC API

The NPL Gateway to ODBC is a tool for Niakwa Developers wishing to access third-party databases from within their Niakwa applications. The NPL Gateway to ODBC consists of the NPL Gateway to ODBC Programmer's Guide and one diskette.

The following describes the contents of the NPL Gateway to ODBC diskette.

BOOT.OBJ	Boot program used by the Niakwa ODBC example programs.
ODBCDEMO.BS2	Diskimage containing ODBC demo and library routines.
ODBCNOTE.TXT	A text file which describes writing ODBC 32-bit applications.
PREBOOT.OBJ	Preboot program used when starting the NPL example programs.
RTIWIN.INI	Sample file with Niakwa ODBC example program entries.
SQL.BS2	Library program diskimage for the NPL Gateway to ODBC.

TEST.BAT Batchfiletoexecutethedemoprogram .

1.3 PRODUCT BENEFITS

The NPL Gateway to ODBC works in conjunction with Microsoft's ODBC SDK. The Microsoft ODBC SDK provides a set of sample drivers and a driver management layer that allows application developers, in this case NPL developers, to access any database that has an ODBC compliant driver written for it. Typically, this means databases that support dynamic SQL, but drivers also exist for many non-relational database formats (i.e., Btrieve, Paradox, Microsoft Access, dBase, Excel, text files etc.). Samples of these drivers are shipped with the Microsoft ODBC SDK, allowing programmers to test their code against popular type of databases.

NOTE: If your system is not equipped with an ODBC driver resource, the file DATAACC.EXE is available on the Windows 95 SDK and Office 97 CD's. This file will install sample 32-bit ODBC drivers on your system.

The NPL Gateway to ODBC interface also allows developers to support multiple back-end databases with the same source code. The "connectivity" part of the Microsoft ODBC API ensures that your application can get at the data that it needs, no matter which driver it needs to go through to get at it.

In addition to the above, ODBC compliant applications provide the following benefits:

ODBC compliance is achieved, thus satisfying a requirement requested by many corporate accounts.

Access to non-PC based data files is easily achieved through SQL servers (i.e., NetWare/SQL, SQL/400 etc.).

Used appropriately in a client/server environment, ODBC can substantially improve performance and reliability while reducing network traffic and minimizing workstation memory requirements.

Extensive SQL commands are available to access data.

1.4 PRODUCT CONSIDERATIONS

Although ODBC compliant NPL applications can have many benefits, there are some issues the developer must consider before adapting their applications.

Non-portable This product is for use with MS-Windows 95/NT applications only.

Niakwa's Gateway to ODBC is different than our portable Niakwa Data

Manager (NDM). As such, NPL source code will need separate routines to handle each.

Prerequisite knowledge

Use of the NPL Gateway to ODBC requires the programmer to acquire an understanding of operation of Microsoft's ODBC SDK and its supplied drivers, as well as a basic understanding of SQL.

Performance

Although performance is good for most API's, the NPL Gateway to ODBC adds a third-level routing call before a database driver can be executed (i.e., Niakwa's Gateway API, Microsoft's ODBC API and the database driver itself).

By default, ODBC works synchronously. This means that a function call does not return until the operation it specifies is complete. However, some database drivers also support asynchronous type calls, although these do not use the MS-Windows event queue.

NOTE: This product requires RTIWIN32 or RTPWIN32 version 5.00 or greater of the NPL Release V for MS-Windows .

1.5 PRODUCT CONCEPTS

The NPL Gateway to ODBC addresses the needs of NPL developers whose markets are forcing ODBC compliance upon their applications. In addition, this product also allows NPL developers to support multiple databases without having to leave the NPL environment.

This product also provides a strategic option allowing for ODBC compliance when it is necessary.

This page intentionally left blank

CHAPTER 2
INSTALLATION

2.1 OVERVIEW

This chapter provides instructions for installing the NPL Gateway to ODBC.

Section 2.2 discusses the system requirements for the NPL Gateway to ODBC.

Section 2.3 discusses configuration requirements for the NPL Gateway to ODBC.

Section 2.4 discusses installing the NPL Gateway to ODBC.

2.2 NPL GATEWAY TO ODBC SYSTEM REQUIREMENTS

The NPL Gateway to ODBC is designed to operate on systems meeting the following requirements:

MS-Windows 95 or greater.

MS-Windows NT 3.51 or greater.

A NPL Release V Development Package for Windows.

A NPL Release V MS-Windows RunTime Package (Revision 5.00 or greater).

NOTE: NPL Release V Revision 5.00 or greater is required. Contact your authorized Niakwa Distributor for information on updating to the current NPL Revision for MS-Windows.

A Microsoft ODBC 3.0 SDK.

2.3 NPL GATEWAY TO ODBC CONFIGURATION REQUIREMENTS

This section discusses the specific configuration requirements for the NPL Gateway to ODBC.

2.3.1 MS-Windows 95/NT Configuration Requirements

MS-Windows 95 or greater and Microsoft's ODBC 3.0 SDK are required. All other Windows configuration requirements for the NPL Gateway to ODBC are the same as documented in the Windows Addendum of the NPL Release V MS-DOS Supplement.

2.4 INSTALLING THE NPL GATEWAY TO ODBC

This section discusses the installation of the NPL Gateway to ODBC. The NPL Gateway to ODBC API consists of one (1) diskette labeled:

NPL Gateway to ODBC - Disk 1 of 1

NOTE: The MS-Windows RunTime and Development Package must be installed before the NPL Gateway to ODBC. Refer to Chapter 2 of the NPL Release V MS-DOS Supplement and Chapter 2 of the NPL MS-Windows Addendum for details.

2.4.1 Installing the NPL Gateway to ODBC Files

The NPL Gateway to ODBC library offers developers with a SQL background a direct approach to implementing ODBC support into their applications. The ODBC API library contains a series of SQL Core, Level 1 and Level 2 function calls based upon the ODBC 3.0 specification.

To install the NPL Gateway to ODBC Library:

1. Make sure the version 5.00 or later RTIWIN32.EXE is installed.

2. Create and select a target directory for the Gateway to ODBC files:

```
C:\>MD NPLODBC
C:\>CD NPLODBC
```

3. Copy all files from the diskette labeled “NPL Gateway to ODBC”:

```
C:\>XCOPY A:*.* /S
```

4. Set the following environment variables (before running Windows), by adding these lines to the CONFIG.SYS (Windows 95) or using the Environment tab of the System Control Panel applet (Windows NT):

   SET ODBC=C:\NPLODBC
   SET NIAKWA_PREBOOT=C:\NPLODBC\PREBOOT

   The second option is only required if you want to use the PREBOOT.OBJ supplied with the examples. If you already use a preboot program, you may wish to incorporate some of the options in this program into your preboot program.

5. Edit the rtiwin.ini file in the odbcdemo directory, replacing all occurrences of:

   %odbcdemo%

   with the full name of the directory where the files were installed. (eg C:\NPLODBC).

   Append this file to the rtiwin.ini file in your windows directory. eg.:

   copy C:\windows\rtiwin.ini+C:\NPLODBC\rtiwin.ini C:\windows\rtiwin.ini

6. Start windows, and create a shortcut to run the boot program in the demonstration directory, as:

   RTIWIN32.EXE C:\NPLODBC\boot.obj

This page intentionally left blank

CHAPTER 3
WRITING ODBC COMPLIANT APPLICATIONS IN NPL

3.1 OVERVIEW

This chapter discusses programming considerations for writing ODBC compliant NPL applications.

Section 3.2 discusses starting the RunTime with the NPL Gateway to ODBC .

Section 3.3 discusses API naming differences between C and NPL.

Section 3.4 discusses data type equivalences between C and NPL.

Section 3.5 discusses startup and shutdown considerations.

Section 3.6 discusses asynchronous programming considerations.

Section 3.7 discusses API references and how they correspond between C and NPL.

Section 3.8 discusses site specific considerations.

Section 3.9 discusses obtaining information regarding SQL drivers.

3.2 STARTING THE RUNTIME WITH THE NPL GATEWAY TO ODBC

The following section discusses starting the MS-Windows RunTime with the NPL Gateway to ODBC.

3.2.1 External Library DLLCALLX.DLL

Older applications using the 16-bit NPL Gateway to ODBC required the DLLCALLX.DLL external library. This is configured using the /X RunTime startup option on the command line or by using the **ExternalLibraryx=** parameter.

The 32-bit RunTime does not require an external library to access ODBC. No /X startup option is required to use the library (refer to Section 4.7 or Section 3.4 respectively of the NPL MS-Windows Addendum for details on RunTime startup options and RTIWIN.INI parameters).

3.2.2 SQLDEV Program File

All applications which use the Niakwa ODBC library must define an NPL program file "SQLDEV" on the default diskimage. This program file must define the public variable _SQLDEV so that it can be used to locate associated modules of the library. In the ODBCDEMO examples, the SQLDEV program appears as:

```
0010    ;SQLDEV
0020    DIM /PUBLIC _SQLDEV=2
0030    PROCEDURE 'SetDevice/MAIN
        : SELECT #_SQLDEV/D13
        : $DEVICE(#_SQLDEV)="SQL.BS2"
        : END PROCEDURE
```

Here the /MAIN procedure (always executed when the module is INCLUDEd) ensures that device #2 is configured to locate the ODBC API library, and that _SQLDEV is set to 2. The SQL.BS2 library is assumed to be in the current directory of each application.

3.2.3 Special Concerns Regarding Symbols with Underlines

Under NPL, the ASCII underline character displays as a left arrow when using the standard NPL character set. To enter symbols such as _SQL_ERROR under MS-Windows, the default keyboard table must be changed to allow entry of the ANSI underline character (otherwise the underline key underlines the character above the cursor). This change is best done in either the boot program or in a pre-boot program, with the following code fragment:

```
;AnsiUnderlines
DIM S$256,K$32+256+32+256
S$=$SCREEN
STR(S$,VAL(HEX(5F))+1,1)=HEX(5F)      :       ;underline fix
$SCREEN=S$
K$=$KEYBOARD
STR(K$,1+32+VAL(HEX(5F)),1)=HEX(5F)   :       ;underline generates HEX(5F)
STR(K$,1+VAL(HEX(5F))/8,1)=AND HEX(7F) :      ;not a special function key
$KEYBOARD=K$
```

NOTE: The ODBCDEMO programs include a PREBOOT.OBJ program which makes the above changes (among others). Refer to Section 2.4.8 of the NPL Release IV Programmers Guide for details on the PREBOOT option.

3.3 API NAME EQUIVALENCES

All API names and constants defined in the "sql.h" and "sqlext.h" files are also defined in the NPL "SQLH" and "SQLEXTH" include modules.

Names of sql.h and sqlext.h constants are defined prefixed by an underscore, to prevent accidental modification by a program that includes them. For example:

C API Constant Macro	NPL Equivalent API Constant Variable
SQL_ERROR	_SQL_ERROR
SQL_SUCCESS	_SQL_SUCCESS

A number of functions useful for handling language differences between C and NPL are in the "CSTRING" include module. Basic C type structures used by the ODBC API to conform to the WIN32 API are included in the "BASETY32" NPL module.

Where module names in SQL.BS2 are the same as those of modules in the Niakwa Gateway to Windows API, the modules perform the same purpose and in general will be the same code on concurrent releases of these products.

3.4 DATA TYPE EQUIVALENTS

This section discusses data type equivalences between C and NPL.

3.4.1 Structures and RECORDS

References to typedef'd structures are replaced with strings of length #RECORDLENGTH(structure_name). Where member references are undecorated (i.e., no attempt to tie the member to the specific structure with a prefix) all NPL fields are decorated with the full structure name and a '_'.

For example:

```
[in sqlext.h:]
typedef struct tagDATE_STRUCT
        {
        SWORD year;
        UWORD month;
        UWORD day;
        } DATE_STRUCT;
```

```
DATE_STRUCT ds;
ds.year=1994;
ds.month=12;
ds.day=31;
```

becomes:

```
[in NPL SQLEXTH library]
1002 RECORD DATE_STRUCT
:       FIELD date_struct_year=_SWORD$,
:       date_struct_month=_UWORD$,
:       date_struct_day=_UWORD$
:       END RECORD

: DIM ds$#RECORDLENGTH(DATE_STRUCT)
: ds$.date_struct_year=1994
: ds$.date_struct_month=12
: ds$.date_struct_day=31
```

3.4.2 Type Conversion

Most C variables which are not structures or pointers are replaced with NPL numerics, except where these are clearly arrays of characters or strings of a fixed length, in which case they become strings.

API's that require a C (null-terminated) string must be passed either an NPL string (trailing spaces are ignored) or a string which has been manually modified to contain the HEX(00) at the end or has been created using the ‘Cstring() function.

Most ODBC functions that require string arguments do not require them to be null terminated and take a second parameter which indicates the string length.

For example:

```
ignore='SQLExecDirect(hstmt, sqlstr$, LEN(sqlstr$))
```

3.4.3 Pointer Types

All types of pointers require special handling. Where a parameter to a function is a pointer to any kind of structure or non-numeric variable, a string argument must be supplied. In calls where the address of a numeric value which will be used AFTER the call returns is required, a string argument is also required.

Example:

(1) Typical use of pointer to return information immediately:

```
DIM nresultcols
ignore='SQLNumResultCols(hstmt, nresultcols)
if nresultcols = 0
```

(2) Case where pointer will return numeric information after another function call:

```
;last argument of SQLBindParameter is a pointer to a numeric SDWORD
;which is not used until a subsequent 'SQLFetch needs it.

DIM outlen$(_MAXCOLS)#RECORDLENGTH(SDWORD_) :;buffers for fetched column
lengths
ignore='SQLBindCol(hstmt,
        i,
        _SQL_C_CHAR,
        STR(data_$,data_(i)),
        collen(i)+1,
        outlen$(i))
;later ...
rc = 'SQLFetch(hstmt)     :;sets outlen$(i) to column length
                          ;and puts data in data_$ variable
FOR I = 1 TO nresultcols
        if outlen$(i).SDWORD_ = _SQL_NULL_DATA
                ;value of the column is the special NULL value
        str(data_$,data_(i),5)='Cstring$("NULL")
ELSE
        if outlen$(i).SDWORD_ > collen(i)+1
        ;keep track of truncated column values
PRINT TO errmsg$;outlen$(i).SDWORD_ - (collen(i)+1);"chars truncated, col";i-1
        end if
END IF
        ;display the column value
        'printfield(collen(i), STR(data_$,data_(i)))
NEXT I
```

3.4.4 Type Casting

Type casts are not required by the ODBC API.

3.4.5 Return Values

For all ODBC API functions, return values indicate the success or failure of the operation (these are frequently ignored by C code). Under NPL, it is strongly advised to always check a function's return value. While you are permitted to do something like:

```
'SQLTransact(henv, odbc, _SQL_COMMIT)          ;;ignore errors
```

It is clearly better to do something like:

```
ignore='SQLTransact(henv, hdbc, _SQL_COMMIT)
```

so that in the event of an unanticipated error, the error code could at least be checked and displayed (when debugging).

Another possibility would be to implement an 'AssertSuccess(expression) procedure to check that unexpected conditions don't occur (return codes other than _SQL_SUCCESS would result in a diagnostic).

3.4.6 Callbacks

The ODBC API does not require any callback routines.

3.4.7 Name Conflicts

There are no ODBC API calls that use identifiers which are reserved words for NPL.

3.4.8 Dealing with Integers

Most C integer expressions do not require any conversion when translating to NPL. However, there are a few exceptions that should be addressed:

Division (/) — Integer division typically requires an INT( ) function applied to ensure that truncation is enforced. If the expression could be negative, FIX( ) might be more appropriate to ensure rounding toward zero.

OR (|) — Where this is used to construct an options flag from components, + - can be used, but beware using macros that refer to the same bits or combinations of bits that are not mutually exclusive. The "CMACRO" library function 'BitSet(X,Y) returns the correct value for a bitwise X|Y where there is doubt about non-exclusivity.

AND (&) — Where this is used to test an options bit in a flags word, the "CMACRO" library

function 'BitTest(X,Y) returns the correct value for a bitwise X&Y.

NOT (!) The negation of a Boolean value x. If x is known to be either _TRUE(1) or _FALSE(0) the negation can be obtained as 1-x. If the value is not known to be strictly _TRUE or _FALSE, the "CMACRO" library function 'BoolNot(X) returns the correct value.

3.4.9 Pointer Classifications

The 32-bit ODBC API for NPL uses flat 32-bit pointers exclusively.

3.5 STARTUP AND SHUTDOWN REQUIREMENTS

The connection sequence for ODBC 3.0 is more user friendly. Instead of using SQLConnect (requiring server, user id, and password), the recommended procedure is to use QLDriverConnect to obtain a validated hdbc. This call will pop up various driver dialogs until any missing information is filled in.

The complete specification for the source (DSN) is returned in a string, and this could be used on future calls to SQLDriverConnect without user intervention. It can be quite a complicated specification (1K buffer minimum is advised).

Some ODBC functions (notably, 'SQLBindCol and any asynchronous operations) will accept as arguments parameters whose address must remain constant for an unspecified duration after the call returns. By default, NPL's memory management allows it to move variables and code to coalesce free space. This means that the address of NPL variables passed to ODBC by reference can subsequently change, making the old address invalid. If the ODBC libraries subsequently use the invalid saved address, this will result in corruption of the workspace and (most likely) a system crash.

NOTE: It is very important that NPL's memory reorganization must be disabled by any application using the ODBC API. If $OPTIONS byte 47 bit HEX(01) is set, this will disable NPL memory reorganization. The API call 'DisableNPLMemoryReorganize is provided to set this bit properly for ODBC applications.

In addition, ODBC applications must ensure that variables passed to ODBC by reference remain declared for the required duration. In general, this means that variables bound to columns (especially for asynchronous operations) should not be allocated as /RECURSIVE variables (explicitly or implicitly).

3.5.1 Cleanup Responsibilities

The careful release of ODBC resources is strongly recommended to allow an application to be rerun during the testing phase. For example, any environment handle allocated using 'SQLAllocEnv should be released via 'SQLFreeEnv.

In the examples, the 'GeneralCleanup routine is defined to ensure that important ODBC resources are released. This routine is flagged as the /EXIT procedure to ensure that it is run whenever the program code is deresolved. Handles or identifiers of resources that may need cleanup are stored in module /STATIC variables so that they can be accessed by the cleanup procedure as well as the creator. When resources are released, the variables are set to an appropriate value for "not allocated" (usually 0), to ensure that they are never released twice (some ODBC API routines will crash if this is done.)

3.6 ASYNCHRONOUS PROGRAMMING USING THE NIAKWA ODBC API LIBRARY

Programmers should be aware that, by default, function requests that may take a substantial amount of time to complete effectively leave the windows interface unresponsive while the call is in progress. This occurs because the function call does not return until the operation is complete (the driver operates 'synchronously') and no polling of the windows message queue occurs during the operation.

Some ODBC drivers also support 'asynchronous' execution of certain functions, using the _SQL_ASYNC_ENABLE option of 'SQLSetStmtOption. If asynchronous execution has been requested for a function, the function call returns immediately with a return code (_SQL_STILL_EXECUTING) indicating that the function call is not yet complete. The function call must be repeated periodically until a return code indicates that the operation is complete (or failed). During the operation, the application can perform other work, including explicitly polling the windows message queue (via $BREAK) to keep the interface responsive. Note that the test drivers shipped with the Microsoft ODBC execute locally, and do not support asynchronous operations.

An NPL application that does this kind of asynchronous operation should disable the ability to close the main NPL application window, or otherwise take precautions to ensure that pending operations are canceled, and other ODBC resources are properly released in the event that the operator becomes impatient and attempts to cancel the operation before it has completed.

The ODBC API for NPL is not event driven, but the API calls can be used in conjunction with event driven programs that use the Visual NPL. When used in conjunction with event driven programs, the use of the asynchronous option will require setting some kind of periodic timer to check for the completion of the operation. Current API specifications (ODBC 3.0) do not provide for message notification of completed asynchronous requests.

3.7 API REFERENCE

The specifications for the API calls closely follows the C version. Where a parameter is passed by value, usually a numeric is used. Where a value is passed by reference, either a numeric or a string value is used as required - unless the numeric reference will be used on a later call.

Where numeric values are returned in later calls, the required argument is a RECORD of the appropriate type, and the numeric value must be explicitly extracted for subsequent use.

For example:

In C:

```
RETCODE SQLAllocEnv( HENV FAR* henv)
/* henv is not used as input */


/* puts the environment handle in 'henv' (a HENV type variable). */
/* example: */
HENV henv;
rc=SQLAllocEnv(&henv);
```

In NPL:

```
FUNCTION 'SQLAllocEnv(henv)
; returns the 'henv' value
DIM henv
rc='SQLAllocEnv(henv)    ;value returned in henv immediately
```

The API specification is usually quite specific about whether each parameter is input or output to the call.

When passing strings to the ODBC API's, an explicit string length is typically required also. C programs may pass the special value _SQL_NTS if the string is null-terminated. NPL programs may prefer to use a LEN(x) function to avoid the need to put a null in the string.

3.8 NAMES OF SERVERS, USERS AND PASSWORDS

This is site-specific information, and you will need to know the correct values for your site. The dialog boxes displayed by the 'SQLDriverConnect will guide you to the appropriate data source. The ODBC 3.0 SDK installs a number of data sources. blank user id and password is all that is required for these.

The examples (module "SERVLIST") show how the names of available sources can be queried from the environment via the 'SQLDriverConnect call.

3.9 SQL Commands

Most of the tasks that can be done with ODBC requires a basic knowledge of SQL. The Microsoft ODBC 3.0 SDK does not come with any kind of on-line syntax charts or explanations of SQL semantics and terminology. The printed or CD-ROM version of Microsoft's ODBC 3.0 Programmers Reference contains syntax diagrams for the recommended syntax for ODBC drivers, but the semantics are not explained.

In general, this information must be obtained from the ODBC driver vendor or general purpose SQL tutorials.

CHAPTER 4
ODBC FUNCTIONS

4.1 OVERVIEW

This chapter categorizes the ODBC functions supported by the NPL Gateway to ODBC. All ODBC functions supported are defined in Chapter 22 "Function Summary" in the Microsoft ODBC 3.0 Programmers Reference manual.

Section 4.2 categorizes the ODBC functions supported by the NPL Gateway to ODBC.

4.2 STANDARD ODBC FUNCTION CATEGORIES

This section summarizes the supported ODBC function categories and associated functions supported by the NPL Gateway to ODBC. For more information about conformance designations, refer to "ODBC Conformance Levels" in Chapter 1, "ODBC Theory of Operation" in the Microsoft ODBC 3.0 Programmer's Reference. For more information about the syntax and semantics of each function supported, refer to Chapter 22, "Function Summary" in the Microsoft ODBC 3.0 Programmer's Reference.

4.2.1 Supported ODBC Categories and Functions

This section outlines the various categories of ODBC calls and the functions associated with them. As noted above, a complete description of the syntax and semantics of each supported function can be found in Chapter 22 “Function Summary” in the Microsoft ODBC SDK 3.0 Programmers Reference.

The NPL Gateway to ODBC API library is based on the ODBC 3.0 specification. Some functionality has changed since the 16-bit version, which was based on the 2.01 specification. All SQLxxx FUNCTIONs now use $DECLARE interfaces directly. This reduces the amount of interface code, but beware:

> Parameter range checking is less strict with $DECLARE.
>
> The return code of most SQL calls need not be saved (except the binding calls, below). It is strongly recommended that you do so to assist in debugging.

In most function calls, numeric values that were previously returned in strings and were extracted using FIELD specifications, now require a numeric parameter, and return a value directly into the variable.

In particular:

The SQLAllocxxx calls to allocate handles now return the handle directly into a numeric parameter. No intermediate string variable is required (or permitted).

The SQLDataSources call returns the string length parameters directly into numeric variables.

The SQLError call returns the native error code and error lengths parameters directly into numeric variables.

NOTE: An important exception to this: When binding columns or parameters, using the following functions:

SqlBindCol
SqlBindParam
SqlBindParameter
SqlSetDescRec

The variables used to receive column lengths must still be (4-byte) string variables, and the column length value is extracted using the .SWORD field format after the SQLFetch. Similarly, the variables used to receive column values must be string variables, even if the column type is numeric. The field must be converted to a numeric after the SQLFetch operation.

The ODBC functions provided in the NPL Gateway to ODBC can be classified into nine categories. The following lists each category and its associated functions.

Connecting to a Data Source
The functions associated with connecting to a data source include:

SQLAllocEnv
SQLAllocConnect
SQLConnect
SQLDriverConnect
SQLBrowseConnect

Obtaining Information about a Driver and Data Source
The functions associated with obtaining information about drivers and data sources include:

SQLDataSources
SQLDrivers
SQLGetInfo
SQLGetFunctions
SQLGetTypeInfo

Setting and Retrieving Driver Options
The functions associated with setting and retrieving driver options include:

SQLSetConnectOption
SQLGetConnectionOption
SQLSetStmtOption
SQLGetStmtOption

Preparing SQL Requests
The functions associated with setting up SQL requests include:

SQLAllocStmt
SQLPrepare
SQLBindParameter
SQLParamOptions
SQLGetCursorName
SQLSetCursorName
SQLSetScrollOptions

Submitting Requests
Functions associated with submitting specific requests from a driver include:

SQLExecute
SQLExecDirect
SQLNativeSql
SQLDescribeParam
SQLNumParams
SQLParamData
SQLPutData

Retrieving Results and Information about Results
Functions specific to request results include:

SQLRowCount
SQLNumResultCols
SQLDescribeCol
SQLColAttributes
SQLBindCol
SQLFetch
SQLExtendedFetch
SQLGetData
SQLSetPos
SQLMoreResults
SQLError

Obtaining Information about the Data Source's System Tables (Catalog Functions)
Functions used to determine information regarding data source's system tables include:

SQLColumnPrivileges
SQLColumns
SQLForeignKeys
SQLPrimaryKeys
SQLProcedureColumns
SQLProcedures
SQLSpecialColumns
SQLStatistics
SQLTablePrivleges
SQLTables

Terminating a Statement
The functions used to terminate statements include:

SQLFreeStmt
SQLCancel
SQLTransact

Terminating a Connection
The following functions are used when terminating a connection:

SQLDisconnect
SQLFreeConnect
SQLFreeEnv

CHAPTER 5
NPL GATEWAY TO ODBC DEMOS

5.1 OVERVIEW

This section describes the NPL Gateway to ODBC Demo example programs that are included with the NPL Gateway to ODBC.

The ODBCDEMO examples demonstrate the use of the NPL Gateway to ODBC API library to create and access a sample data table. These examples are loosely based on the C Demo examples in the Microsoft ODBC SDK Programmer's Reference.

Section 5.2 describes the contents of the NPL Gateway to ODBC Demo examples.

Section 5.3 discusses installing the NPL Gateway to ODBC Demo examples.

Section 5.4 discusses configuring the NPL Gateway to ODBC Demo examples.

Section 5.5 discusses adding the NPL Gateway to ODBC Demo example programs to the MS-Windows Program Manager.

Section 5.6 discusses starting the NPL Gateway to ODBC Demo examples.

Section 5.7 discusses the NPL Gateway to ODBC Demo examples.

5.2 CONTENTS OF THE NPL GATEWAY TO ODBC DEMOS

The Demo example programs for the NPL Gateway to ODBC are contained on a single diskette. The following section describes the demo files. Refer to Section 2.4 for details on installing the NPL Gateway to ODBC and example files.

The following describes the example files used by the NPL Gateway to ODBC.

BOOT.OBJ	Boot program used by the Niakwa ODBC example programs.
ODBCDEMO.BS2	Program diskimage containing the Niakwa ODBC example programs.
ODBCDEMO.GRP	Windows group file with pre-defined MS-Windows icon.
PREBOOT.OBJ	Preboot program used when starting the NPL example programs.
RTIWIN.INI	Sample file with Niakwa ODBC example program entries.

SQL.BS2 Library program diskimage for the NPL Gateway to ODBC.

5.3 INSTALLING THE NPL GATEWAY TO ODBC DEMOS

Refer to Section 2.4 for details on installing the NPL Gateway to ODBC files.

5.4 CONFIGURING THE NPL GATEWAY TO ODBC DEMOS

This section discusses the configuration of the NPL Gateway to ODBC example programs. The following assumes the NPL Gateway to ODBC has been copied into the C:\ODBCDEMO directory as described in Section 2.3. In addition, it is assumed a Niakwa MS-Windows RTIWIN32 RunTime Revision 5.00 or greater is properly installed and configured.

NOTE: If the NPL Gateway to ODBC files have been installed in a different drive/directory, be sure to adjust the configuration Demos accordingly.

5.4.1 MS-Windows Environment Variables

The following environment variables must be set before running Windows and executing the Niakwa Gateways to ODBC Demos. It is recommended that these lines be added to your AUTOEXEC.BAT file.

```
SET ODBCDEMO=C:\ODBCDEMO
SET NIAKWA_PREBOOT=C:\ODBCDEMO\PREBOOT.OBJ
```

The second environment variable is required if the PREBOOT.OBJ supplied with the Demos is to be used. If a preboot program is already in use, you may wish to incorporate some of the options in this program into your preboot program.

5.4.2 RTIWIN.INI File

Edit the RTIWIN.INI file in the ODBCDEMO directory, replacing all occurrences of:

%odbcdemo%

with the full name of the directory where the files were installed (i.e., C:\ODBCDEMO).

Append this file to the RTIWIN.INI file in your Windows directory. From a DOS prompt enter:

COPY C:\WINDOWS\RTIWIN.INI+C:\ODBCDEMO\RTIWIN.INI C:\WINDOWS\RTIWIN.INI

5.5 ADDING THE NPL GATEWAY TO ODBC TASKS TO WINDOWS DESKTOP

Niakwa recommends adding a shortcut to the MS-Windows Desktop for the NPL Gateway to ODBC Demos.

Follow the steps below to create a shortcut with MS-Windows.

1. Start Windows 95 or NT.

2. Right click on the desktop and choose ‘New’ then ‘Folder’.

3. Select the Description text box. Enter a description for the new folder. For example, enter:

   NPL Gateway to ODBC Demos

4. Open the Demos folder.

5. Right click within the folder and choose ‘New’ then ‘Shortcut’.

6. In the command line of the shortcut properties, enter the RunTime directory and executable, for example:

   C:\NPL\RTIWIN32.EXE

7. Select the name for the shortcut. Enter the following:

   ODBC for Niakwa DEMO

8. Select OK or press Enter. The Shortcut is created.

9. Open the Properties for the Shortcut, and set the ‘Start In’ directory to the NPL Gateway to ODBC directory.

10. Select ‘Apply’ then ‘OK’.

Refer to Section 5.6 for details on starting the NPL Gateway to ODBC Demos.

5.6 STARTING THE NPL GATEWAY TO ODBC DEMOS

To start the NPL Gateway to ODBC Demos, the following steps are required.

1. From the MS-Windows Desktop, select the NPL Gateway to ODBC Demos Folder.

2. Click on the "ODBC for Niakwa Demo" icon to start the NPL Gateway to ODBC Demos.

5.7 NPL GATEWAY TO ODBC DEMOS

This section discusses the example programs provided with the NPL Gateway to ODBC .

5.7.1 Example 1

Example 1 creates a new table called 'EXAMPLE1' with the following column names and lengths:

CustID	CHAR(5)
Company	CHAR(40)
Address	CHAR(40)
City	CHAR(40)
Region	CHAR(20)
PostalCode	CHAR(20)
Country	CHAR(20)
Phone	CHAR(20)

Unlike the original C example, which uses an 'integer' type, this example avoids using anything but the most basic character data type, so that the example may be used with a variety of ODBC servers.

NOTE: The EXAMPLE1 table is created in the location currently selected (by the ODBC applet in the Control Panel) for the specified server. This location information is specified differently for each ODBC driver. For some ODBC drivers, this will create a new EXAMPLE1.xxx file in a specified directory. For example:

"C:\ODBC2\SMPLDATA\PARADOX\EXAMPLE1.DB".

For others (i.e., MS-Access), it will add a table to an integrated database file.

5.7.2 Example 2

Example 2 allows the execution of an SQL statement against the table set up by Example 1 and displays the results in a columnar format. Typically, an SQL SELECT statement is used for this. Other SQL statements can theoretically be used here. For example, DELETE could be used to get rid of a set of rows, or UPDATE to modify a set of rows (this has not been well tested).

Other statements that require use of the dataset cursor are not suitable for use with this example, since the connection is opened before each statement is executed and closed immediately after.

Other tables known to the server can also be accessed. For the ODBC 2.01 version, each server has 3 tables defined; Customer, Orders and Product. The names of the columns in these tables can be defined by using a "SELECT*FROM table-name" statement.

NOTE: The column-names will appear as headings for the first screen.

The EXAMPLE1 table is a copy of the CUSTOMER table, so the CUSTOMER table should look familiar.

To cancel the rest of the output from a query, enter 'N' or CANCEL at the "More.." prompt.

To make the EXAMPLE1 table disappear, be sure to do a "DROP TABLE EXAMPLE1" as the last command.

The ODBC example halts after a blank request is entered for Example 2.

Code for the Demo examples can be inspected by selecting MODULE "EXAMPLE1" or MODULE "EXAMPLE2".