Contents

1 Application Development on Domino for iSeries .......................... 1
   About this guide .................................. 1
   System Requirements ................................ 1
   Additional resources for Domino application development .......... 2

2 Application Development Considerations ......................... 5
   General considerations for using C, C++, and Java APIs .............. 5
   Obtaining the C, C++ and Java APIs .......................... 5
   Running under the QNOTES user profile ....................... 6
   Threading considerations for OS/400 .......................... 9
   Setting the PATH environment variable .......................... 10
   Starting server add-ins .................................. 11
   Restriction on APIs in exception or cancel handlers ............. 11
   Restriction on using global operators .......................... 11
   Using API handles ..................................... 11
   Teraspace enabling your programs .............................. 12
   ASCII-to-EBCDIC conversion ................................ 12
   ASCII C/C++ Run Time for iSeries PRPQ ........................ 13
   Where to find the header files on iSeries ....................... 14
   Considerations for C APIs .................................. 14
   Compiler requirements for C ................................ 14
   Calling C programs from Domino applications or agents .......... 15
   Ensuring ASCII compilation for C APIs ........................ 16
   Accessing the C header files ................................ 17

   Compiling the C program on iSeries .......................... 17
   Considerations for Java ..................................... 19
   OS/400 software requirements for Java ........................ 20
   Setup required to run Java programs .......................... 20
   Using Java in Domino Agents on iSeries ....................... 20
   Restrictions and requirements for running Java agents ........... 22
   Requirements for compiling Java programs, agents, and servlets .... 22
   JavaMaxHeapSize *NOMAX .................................. 22
   JavaOS400CHKPATH ....................................... 23
   Considerations for LotusScript functions on iSeries ............... 24
   Ways of calling programs from Domino agents ................... 28
   About adding hook drivers .................................. 30

3 Integrating Notes and DB2/UDB for iSeries Data ................. 31
   Setup required to access DB2/UDB for iSeries ................... 32
   About authority when Domino applications access DB2/UDB for iSeries ........ 33
   General requirements and differences for LS:DO and @Db functions .... 37
   Limit on number of concurrent SQL statements .................... 38
   Precaution for prestarted jobs that process SQL requests ........... 38
   Differences for LS:DO ....................................... 38
   Remote connection differences ................................ 41
   Where to find error messages .................................. 43

4 Domino for iSeries APIs ...................................... 45
   Domino for iSeries server APIs ................................ 45
   List Domino Servers (QnninListDominoServers) .................. 46

About authority when Domino applications access DB2/UDB for iSeries
Chapter 1
Application Development on Domino for iSeries

Welcome to the Application Development book for Domino for iSeries. This book supplements the application development information that ships with Domino on all platforms by documenting Domino development differences and considerations unique to iSeries. For more information on additional Application Development resources see, "Additional resources for Domino application development" later in this guide.

About this guide

This guide covers the following information:

- Application Development Considerations for developing on iSeries
- Considerations for using C
- Considerations for using C++
- Considerations for using Java
- Considerations for using LotusScript
- Integrating Notes and DB2/UDB for OS/400 data
- Application Programming Interfaces (APIs) unique to iSeries

System Requirements

To develop Domino applications for your iSeries server you will need the following:

Workstation Platform

- Windows 98, Windows NT 4.0, Windows 2000, or Windows XP
- 130 MB hard drive space

iSeries server requirements

- 60 MB hard drive space
- IBM iSeries ILE C Compiler, available as the Product 5722-WDS Option 52
- QSYSINC - OS/400 System Openness Includes, available through the "Install Licensed Program" Option 13
• RUNDOMCMD - OS/400 Domino Command, ships with Domino 6 for iSeries and available for download at
• ASCII C/C++ Run Time for OS/400, available for download at
  http://www.ibm.com/servers/eserver/iseries/developer/factory/asciirt

Additional resources for Domino application development

The following resources provide additional information for developing for Domino for iSeries:

• Lotus Developer Domain
  The Lotus Developer Domain contains technical information about Lotus software from IBM. The Web site provides discussion forums, a technical journal, code samples, demonstration software as well as complete documentation for all Lotus software. It is available at the following URL:
  http://www.lotus.com/ldd

• Lotus Domino for iSeries Developer’s Tools Web site
  The Domino for iSeries Developer Tools Web site contains tools and sample code that you can use to integrate Domino for iSeries applications with OS/400 applications and data. The Web site is available at the following URL:
  http://www.ibm.com/eserver/iseries/domino/domdevtools.htm

• The iSeries Information Center
  The iSeries Information Center contains hardware, software and application development information in one resource. Domino for iSeries application developers should review information found in the following topics of the Information Center:
  • Programming
    From the Information Center Home page, click Programming. This topic contains links to the WebSphere Development Studio ILE C/C++ Programmer's Guide and the System APIs.
• Integrated File System
  This topic describes the structure and use of the iSeries integrated file system, including rules for specifying path names.
  To access this information in the V5R1 version of the iSeries Information Center, click **Database and file systems->File systems and management**
  To access this information in the V5R2 version of the iSeries Information Center, click **File systems and management->Integrated file system**

• Database
  This topic describes the DB2/UDB for iSeries call level Interface and DB2/UDB for iSeries SQL syntax.
  To access this information in the V5R2 version of the iSeries Information Center, click **Database**.
  To access this information in the V5R1 version of the iSeries Information Center, click **Database and file systems->DB2 UDB for iSeries**

The Information Center ships on CD with every iSeries server and is available online at:
http://www.ibm.com/servers/eserver/iseries/infocenter
Chapter 2
Application Development Considerations

This chapter describes application development considerations unique to Domino for iSeries.

General considerations for using C, C++, and Java APIs

There are several requirements and restrictions that apply to all programs that use the Notes C, C++, or Java APIs.

- Obtaining the C, C++, and Java APIs.
- Run all programs under the QNOTES user profile.
- Be aware of threading considerations on OS/400.
- Define a Path environment variable to identify Domino directories.
- Use a symbolic link to start server add-ins automatically.
- Avoid putting Domino APIs in exception and cancel handlers.
- Avoid using global new() and delete() operators.
- As needed, change your programs to use the iSeries 32-bit API handles.
- Make sure that your applications are Teraspace enabled.
- Make sure you properly convert between EBCDIC and LMBCS (ASCII).

Obtaining the C, C++, and Java APIs

Depending on the APIs you use to develop your Domino applications, you can obtain the Domino C, C++, and Java APIs in different ways:

- The C API runtime is shipped with the base option of the Domino product. The header files and sample programs are shipped with option 1.
- The C++ APIs can be downloaded as part of the Lotus C++ API toolkit from the Lotus Developer Domain Web site or the Lotus Domino for iSeries Developer's Tools Website.
- The Lotus Developer Domain is available at the following URL:
  http://www.lotus.com/ldd
- The Lotus Domino for iSeries Developer's Tools Web site is available at the following URL:
  http://www.ibm.com/eserver/iseries/domino/domdevtools.htm
- The Java APIs ship with the Domino product.

**Running under the QNOTES user profile**

All Domino applications must run under jobs that use the QNOTES user profile. The user profile defines operational characteristics for jobs, and is the source for the special authorities that jobs need. The following processes run under the QNOTES profile when they are started:

- The Domino server
- Programs that the Domino server starts as a result of ServerTasks or ServerTasksAt settings in the NOTES.INI file
- Programs that start as a result of Program documents in the Domino Directory
- Hook drivers and extension manager applications that run in the Domino server
- Agents that run on the Domino server
- Programs that start when you enter the Load command on the Domino server console

If your application is not designed to run in the above processes (such as an interactive application), you must modify your application so that it runs under the QNOTES user profile using one of the following methods:

- Inside your application, switch to the QNOTES profile by using the following system APIs:
  - QSYGETPH - get profile handle
  - QWTSETP - set profile
- Use the Submit Job command (SBMJOB) to submit the job to run under QNOTES.
• Use the RUNDOMCMD utility that ships with Domino 6 for iSeries and is available as a download from the following Web address:


This utility switches to the QNOTES user profile, runs your application, sets up the PATH and current directory correctly and then switches back to the user profile you were using. Using this utility, you do not need to write additional code in your applications.

• Use the SETDOMENV CL command

The following sample sets all the appropriate environment variables for the API program and then submits a call to the API program under the QNOTES userid. Notice the COPYENVVAR parameter in the SBMJOB command, *YES must be specified for the environment variables to be copied into the API program call.

```*/
  /* PROGRAM:  MYPGMCL                      */
  /* */
  /* */
  /* FUNCTION:  FRONT END TO MYPGM NOTES C API PGM */
  /* */
  /* */
  /* INPUT */
  /* */
  /* CHANGE HISTORY: */
  /* */
  /* DATE  WHO       CHANGE DESCRIPTION */
  /* */
  /* 02/18/2000 BEO   INITIAL INSTALL */
  /* */
*/

PGM
  DCL  VAR(&SERVER) TYPE(*CHAR) LEN(10)
  DCL  VAR(&PARTITION) TYPE(*CHAR) LEN(2)
  DCL  VAR(&PATHNAME) TYPE(*CHAR) LEN(30)
  DCL  VAR(&ENTRYTYPE) TYPE(*CHAR) LEN(15)
  DCL  VAR(&PATH) TYPE(*CHAR) LEN(256)

/* TEMP */
  CHGVAR  VAR(&SERVER) VALUE('d27dbt06')
  CHGVAR  VAR(&PARTITION) VALUE('07')
```
/* ADD ENVIRONMENT VARIABLE PATH */

CHGVAR VAR(&PATH) +

VALUE('/qibm/proddata/lotus/notes:/qibm/user+
    rdata/lotus/notes:/notes/' *TCAT
    &SERVER)

ADDENVVAR ENVVAR(PATH) VALUE(&PATH)

MONMSG MSGID(CPFA980) EXEC(CHGENVVAR
ENVVAR(PATH) +
     VALUE(&PATH))

/* ADD ENVIRONMENT VARIABLE PARTITION */

ADDENVVAR ENVVAR(NOTESPARTITION)
VALUE(&PARTITION)

MONMSG MSGID(CPFA980) EXEC(CHGENVVAR +
ENVVAR(NOTESPARTITION)
VALUE(&PARTITION))

/* CHANGE CURRENT DIRECTORY */

CHGCURDIR DIR('/notes/' *TCAT &SERVER)

/* CHANGE CURRENT LIBRARY */

CHGCURLIB CURLIB(QNOTES)

/* CALL PROGRAM */

SBMJOB CMD(CALL PGM(MYLIB/MYPGM) JOB(MYJOB) +
    USER(QNOTES) CURLIB(*USRPRF) +
    INLLIBL(*NONE) CPYENVVAR(*YES)

END: ENDPGM

SBMJOB CMD(CALL PGM(MYLIB/MYPGM) JOB(MYJOB) USER(QNOTES)
CURLIB(*USRPRF) INLLIBL(*NONE) CPYENVVAR(*YES)
Threading considerations for OS/400

Consider the following when developing your Domino for iSeries (OS/400) applications:

The following programs run in processes (jobs) that can have multiple threads:
- Programs that are started as a result of Program documents in the Domino Directory and any programs they call
- Programs that are started as a result of entering the Load command on the Domino server console and any programs they call
- Server-based agents that are called by a Web browser when the NOTES.INI specifies DominoAsynchronizeAgents=1

The following programs run in secondary threads:
- Hook drivers and extension manager applications that run in the Domino server
- C programs that are called by the LotusScript Declare statement in agents and any programs that are called from those C programs
- Programs that are specified in the LotusScript Shell statement in agents and any programs that are called by those programs

The iSeries operating system (OS/400) has restrictions on jobs capable of running multiple threads and on the use of secondary threads. For details, see the Programming topic in the iSeries Information Center at the following URL:

http://www.ibm.com/eserver/iseries/infocenter

Certain Domino APIs require the ability to start another thread (the job it runs in must be multithread capable). Interactive jobs on OS/400 cannot be multi-thread capable.

A program that runs in a *NEW activation group will end a multithreaded job when it returns. For details, see the Programming topic in the iSeries Information Center at the following URL:

http://www.ibm.com/eserver/iseries/infocenter

Stand alone programs MAY need to be enabled to allow multi-threading if calling Domino APIs. Some Domino APIs need to start a second thread. These stand alone programs are assumed to be running in batch mode and not interactively. Interactive jobs on the iSeries currently can not be multi-thread capable.
Setting the PATH environment variable

Addin programs and programs that run under the Domino server already have the PATH environment set up. To run stand-alone programs using C, C++ or Java API programs you must manually set up the PATH environment variable that identifies the following items:

- The Domino executable directory
  The path for this directory is /QIBM/ProdData/Lotus/Notes.
  **Note** This can change from release to release and each Domino server could have a different executable directory. To correctly determine the executable directory for a server, use the Qnnin* APIs to retrieve the executable directory for the server you are running against.

- The user executable directory
- The resource directory
  The path for this directory is /QIBM/UserData/Lotus/Notes.

- The data directory for the Domino server
  You specify the path for the data directory when you set up the Domino server.

You can set up the PATH in one of several ways:

- You can use the QnninSetDomEnv API
- You can use the SETDOMENV CL command
- You can launch your program using the RUNDOMCMD CL Command which sets the PATH
- You can set up the path explicitly by the application as shown below where the server data directory is /NOTES/DATA and the executable directory is /QIBM/ProdData/Lotus/Notes

```plaintext
addenvvar envvar(PATH)
value('/QIBM/PRODDATA/LOTUS/NOTES:/QIBM/USERDATA/LOTUS/NOTES:/NOTES/DATA')
```
Starting server add-ins

To start server add-ins automatically or on the server console, create a symbolic link for the add-in in the /QIBM/UserData/Lotus/Notes directory. For example, for a program named MyAddin that is in the library MyLibrary, specify the following Add Link (ADDLNK) command to create a symbolic link:

```
ADDLNK OBJ('/QSYS.LIB/MYLIBRARY.LIB/MYADDIN.PGM')
NEWLNK('/QIBM/UserData/Lotus/Notes/MYADDIN.PGM')
LNKTYPE(*SYMBOLIC)
```

Once created, the symbolic link will persist even if the program is deleted and created again.

For more information about symbolic links, see the iSeries Integrated File System topic in the iSeries Information Center at the following URL:

http://www.ibm.com/eserver/iseries/infocenter

Restriction on APIs in exception or cancel handlers

Do not put Domino APIs in exception or cancel handlers. The Domino APIs use locks to synchronize access to resources with the server. The result of putting Domino APIs in exception or cancel handlers could range from deadlocking the Domino server to requiring an IPL (restart) of your iSeries server.

If a Domino API is invoked in your exception or cancel handler, a deadlock could occur. If a Domino API in your application is holding a lock when an exception or cancel occurs and the handler issues another Domino API, a deadlock occurs. The effect of the deadlock depends on whether it occurs in an exception handler or a cancel handler:

- If the deadlock occurs in an exception handler, you must end that job and restart the Domino server and any Domino applications that are running on the server.
- If the deadlock occurs in a cancel handler, you need to end that job abnormally. As a result of ending the job abnormally, you must IPL your iSeries server.

Restriction on using global operators

Do not use global new() or delete() operators in any add-in, library, or LSX. If you need to override those operators, confine them to class scope. Using global new() or delete() operators may result in some parts of the Domino server not working correctly. This restriction is related to an OS/400 restriction that there can be only one global new() and delete() operator per activation group.

Using API handles

API handles are 32 bits on iSeries. If your code depends on handles being 16 bits, change your code to accommodate the 32-bit handles.
Teraspace enabling your programs

Teraspace is the term for process-local storage on OS/400. It allows for larger contiguous memory allocations and for the use of 8-byte pointers (instead of 16-byte pointers required for Single Level Store data). Currently, the Domino application makes use of teraspace in certain areas where it needs more than 16MB of continuous storage. Domino is also fully teraspace-enabled. In other words, you can pass teraspace memory to the Domino application and the Domino application can access the larger memory allocations. To ensure your applications are able to run in this environment there are some considerations described below that you should use when creating modules, programs, or service programs.

Note For more information on Teraspace enabling your applications, see Chapter 4 of the iSeries ILE Concepts (SC41-5606) book which is available at: http://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/c4156066.pdf

When you develop Domino applications, consider the following:

- At a minimum, all programs and service programs that use the Domino C/C++ APIs should be teraspace enabled. The modules created using the CRTCMOD or CRTCPPMOD command can be teraspace enabled using the TERASPACE(*YES) command option.
- Using the *INHERIT storage model (STGMDL(*INHERIT) option at compile time) provides the most flexibility in putting those modules into programs and service programs later.
- Using STGMDL(*INHERIT) on service programs (STGMDL(*INHERIT) option CRTSRVPGM time) can allow the service programs to be loaded into SLS or TERASPACE activation groups. If you are using *CALLER for a service program activation group, consider using *INHERIT for your service program storage model.

For the latest information on teraspace enabling your Domino applications, see the Domino for iSeries C and C++ toolkits, which can be downloaded from the Lotus Developer Domain at:

http://www.lotus.com/ldd

ASCII-to-EBCDIC conversion

Notes stores data internally in LMBCS (Lotus Multi-Byte Character Set), and the character string representation for the Notes C API parameters is assumed to be LMBCS. When you write C programs using EBCDIC character strings, you must use the Notes OSTranslate API to convert between the EBCDIC strings used by the program and the LMBCS parameters passed to and from the C APIs.
You may see Notes sample programs where this is not done. Those sample programs are technically incorrect, although functional in most cases. The sample programs usually work because ASCII code page 850 (PC ASCII) is a proper subset of LMBCS; therefore, characters in that character set do not need to be translated to and from LMBCS because they are already in LMBCS. Because most of today's Notes applications use code page 850, or its 7-bit ASCII subset, the use of OSTRanslate is not needed under those circumstances.

The C++ APIs require EBCDIC strings for input and output, rather than the LMBCS strings used for C APIs.

**ASCII C/C++ Run Time for iSeries PRPQ**

The ASCII C/C++ Run Time for iSeries PRPQ provides a seamless solution for conversion between an ASCII application and the EBCDIC ILE environment of Domino for iSeries. ASCII C/C++ Run Time serves as an ASCII-EBCDIC interface layer for commonly used system APIs, such as printf() or scanf(), that take and return EBCDIC strings. You need to use such an interface layer in cases where user-written code is compiled into ASCII, such as when using the Domino C APIs. By using ASCII C/C++ Run Time, you can significantly reduce the number of changes required when porting Domino applications to your iSeries server.

Besides ASCII/EBCDIC conversion, ASCII C/C++ Run Time also supports ASCII/UNICODE conversion.

ASCII C/C++ Run Time is available at the following Web site:

http://www.ibm.com/eserver/iseries/developer/factory/asciirt
Where to find the header files on iSeries

When you install the application development software, the files are loaded into specific OS/400 libraries. The following table shows the OS/400 library in which the tools support is installed and the path for accessing that library through the integrated file system.

<table>
<thead>
<tr>
<th>Item</th>
<th>Library</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>C APIs</td>
<td>QNOTESAPI</td>
<td>/qsys.lib/qnotesapi.lib</td>
</tr>
<tr>
<td>C++ APIs</td>
<td>QNOTESCPP</td>
<td>/qsys.lib/qnotescpp.lib</td>
</tr>
<tr>
<td>LotusScript</td>
<td>QNOTESLSKT</td>
<td>/qsys.lib/qnoteslskt.lib</td>
</tr>
<tr>
<td>Extensions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Considerations for C APIs

Be aware of the following considerations as you use the C APIs.

- Compiler requirements
- Ensuring compilation in ASCII
- Accessing the header files
- Compiling on iSeries

Compiler requirements for C

You can compile the source as a member of a physical file or as an IFS file.

Compiling the source as a member of a physical file

You can compile the C APIs using the OS/400 C compiler on iSeries. If you compile the C application program with the source code as a member of a physical file, the ILE C/C++ compiler can access the headers as they are installed when you add the QNOTESAPI or QNOTESCPP to your library list. No additional setup is needed.
Compiling the source as an IFS file
If you compile the C application programs with the source code as an IFS file, the ILE C/C++ compiler resolves the header files (e.g. defined as #include "inc_name" ) to IFS files (e.g. [hpp_search_path]/inc_name, where the hpp_search_path is defined by the compiler option INCDIR). So you must ensure the C headers are located as IFS files. There are two ways to do this:

- Create a symbolic link from the integrated file system to the actual location of the header files in Library QNOTESAPI or QNOTESCPP.
- Copy the headers from the H/HPP file in the library QNOTESAPI or QNOTESCPP to the integrated file system.

Calling C programs from Domino applications or agents
There are several ways for Domino applications or agents to invoke programs running on iSeries servers. C functions can be called within service programs by using the LotusScript "Declare Function" statement. The routines within the called service program can call other programs. Note that the program being called is running in a thread-enabled job. If the agent is being initiated through the Web server, the program is running within a thread. Depending on the release of OS/400, there are restrictions on the system about what can run in thread-enabled and threaded processes. For example, RPG or COBOL programs could not be called from threaded jobs prior to V4R4 because RPG and COBOL were not yet thread-safe. This restriction has been lifted for OS/400 version V4R4 and later. Calling C routines currently gives the best support for parameter passing. The C programming language will produce thread-safe code by avoiding certain programming constructs. Here is a LotusScript statement that shows how to declare a routine "runthis" which is in the "CmdShell" C service program (*SRVPGM):

Declare Function runthis Lib "/qsys.lib/tenner.lib/CmdShell.srvpgm" (Byval cmdstr As String) As Integer

The runthis routine takes one parameter — cmdstr.

Note For more information, see the Domino Designer Help database -> Index view -> External declarations.
To call the runthis routine in the LotusScript code, simply code runthis("parameters_go_here") where "parameters_go_here" are the parameters being used.

cmdstring="input string"
rc=runthis(cmdstring)
Print "Cmdstring: ";cmdstring

The C program would look like the following:

```c
#include <stdlib.h>

/* This simple program changes the string passed in on cmd to the text 'done calling' */
int runthis(char *cmd)
{
    int rval;
    strcpy(cmd,"done calling");
    return rval;
}
```

As shown here, parameters can be used for both input and output. The string type is unique in that it is passed as a pointer. For additional information see the Calling external C language functions topic in Domino Designer Help database (help6_designer.nsf).

Finally, for parameters other than pointers, _System linkage should be indicated on the routine. This is only supported by the ILE C compiler. For example:

```c
int _System runthis(char *cmd) {
    ... /* your routine code goes here */
}
```

**Ensuring ASCII compilation for C APIs**

Notes C APIs expect and return LMBCS (ASCII) strings, literals, and so on. To get ASCII compilation, you need to use the following compiler flag:

```
/AScp850
```

Since the C++ APIs require EBCDIC and not ASCII, avoid mixing the two.

For more information on ASCII to EBCDIC conversion, see "ASCII-to-EBCDIC conversion," earlier in this chapter.
Accessing the C header files

When you install the C API option on iSeries, the C header files are installed in the QNOTESAPI library as members of the OS/400 file H.FILE. The installation also creates symbolic links in the OS/400 Integrated File System to the header file members as /qibm/proddata/lotus/notesapi/include/*.h.

Within the QNOTESAPI library, the header files are (QNOTESAPI/H). If you compile the C applications programs with the source code as a member of a physical file, then the ILE compiler can access the headers as they are installed. You only need to add the QNOTESAPI to your library list.

If you compile the C applications in the Integrated File System, the compiler will access the headers through by using the symbolic links in the /qibm/proddata/lotus/notesapi/include directory. You only need to specify the directory with the INCDIR compiler command parameter.

Compiling the C program on iSeries

You can link and compile the C program in an OS/400 physical file or compile and link the C application with the source in the Integrated File System.

Compiling and linking the C program in an OS/400 physical file

The following procedures provide list the necessary steps for compiling the application on iSeries. Detailed information on these procedures can be found in the WebSphere Development Studio ILE C/C++ Programmer's Guide.

1. Put the QNOTESAPI library in your library list. For example, enter this OS/400 command:
   
   addlible qnotesapi

2. Specify a define name of OS400 when you compile the C module on iSeries (CRTCMPD command). For example:

   crtcmod module(ctest/foo) define(os400)

   This command creates a module called FOO.MODULE in the library CTEST.
3. After compiling the C module, link the module into a program (CRTPGM command) or a service program (CRTSRVPGM command) on the iSeries on which you compiled the module. When you use either the CRTPGM or the CRTSRVPGM command, bind in the LIBNOTES service program found in the QNOTES library. Here is a link statement that uses CRTPGM to make a program from the above compilation:

```
crtpgm pgm(ctest/foo) module(ctest/foo)
bindsrvpgm(qnotes/libnotes)
```

The CRTPGM command creates FOO.PGM in the library CTEST.LIB.

Depending on what type of Notes application is being written, you may need to link in either or both of the NOTES0 and NOTESAI0 modules. These two modules are in the QNOTESAPI library. To link in either or both of these modules, add the module or modules to the module list of the CRTPGM or CRTSRVPGM command. For example:

```
crtpgm pgm(ctest/foo) module(ctest/foo qnotesapi/notes0)
bindsrvpgm(qnotes/libnotes)
```

4. If the application is to run under the Domino server, you must manually create a symbolic link from the Domino server UserData directory to the location of the executable:

```
addlnk obj('/qsys.lib/ctest.lib/foo.pgm')
newlnk('/Qibm/UserData/Lotus/Notes/foo.pgm')
lntktype(*symbolic)
```

You can run the program by using the OS/400 CALL command. The program must run under the QNOTES user profile and you must set up a PATH environment variable to identify the required OS/400 directories. You can swap to the QNOTES user profile in your program or you can specify QNOTES when you run the program.

The OS/400 Submit Job (SBMJOB) command allows you to specify a user profile when you run a program. Here is an example of using the SBMJOB command to run the FOO program that is in the CTEST library:

```
sbmjob cmd(call pgm(ctest/foo)) user(qnotes)
cpyenvvar(*yes)
```

In addition to specifying the QNOTES user profile, this command copies the environment variable to the submitted job.
Compiling and linking the Domino C application with the source as an IFS file

The following procedures list the necessary steps for compiling the application on iSeries. Detailed information on these procedures can be found in the *WebSphere Development Studio ILE C/C++ Programmer's Guide*.

The commands in IFS section use /QIBM/PRODDATA/QADRT/INCLUDE directory and QADRT/QADRTNTS service program. These depend on installation of the ASCII C/C++ Run Time.

1. Compile and link the Domino C application with the source as a IFS file /ntcapi/csampample/cssample.cpp:

```
CRTCMOD MODULE(NTCAPI/CSAMPLE)
SRCSTMF('NTCAPI/CSAMPLE/CSAMPLE.C')SYSIFCOPT(*IFSIO)
DEFINE(OS400 UNIX)INCDIR('/QIBM/PRODDATA/QADRT/INCLUDE'
' /QIBM/PRODDATA/LOTUS/NOTESAPI/INCLUDE')
```

This compiles the CSAMPLE module in the NTCAPI library using includes in the QADRT and NOTESAPI directory.

2. Create the ILE program executable:

```
CRTPGM
PGM(NTCAPI/CSAMPLE)MODULE(*PGM)ENTMOD(QNOTESAPI/NOTES0)
BNDSRVPGM(QNOTES/LIBNOTES QADRT/QADRTNTS) + DETAIL(*BASIC)
```

After compiling the C module, link the module into a program (CRTPGM command) or a service program (CRTSRVPGM command) on the iSeries on which you compiled the module. When you use either the CRTPGM or the CRTSRVPGM command, bind in the LIBNOTES service program found in the QNOTES library. Here is a link statement that uses CRTPGM to make a program from the above compilation:

```
crtpgm pgm(ntcapi/CSAMPLE) module(NTCAPI/CSAMPLE)
bndsrvgpm(qnotes/libnotes)
```

3. If the application is to run under the Domino server, create a symbolic link from the Domino server Userdata directory to the executable program:

```
ADDLNK OBJ('/QSYS.LIB/NTCAPI.LIB/CSAMPLE.PGM)
NEWLNK('/QIBM/UserData/Lotus/Notes/cssample.pgm')LNKTYPE(*SYMBOLIC)
```

For additional information, see "Additional resources for Domino application development" in Chapter 1.

Considerations for Java

Be aware of the following considerations as you develop applications that use Java APIs or run Java agents:

Chapter 2: Application Development Considerations  19
OS/400 software required to run Java programs or agents
Setup required to run Java programs
Requirements for compiling Java programs, agents, and servlets
Restrictions and requirements for running Java agents
Restrictions for creating Java applications

**OS/400 software requirements for Java**

To run Java programs or Java agents on iSeries, you must install following software:

- OS/400 - Qshell Interpreter option, 5722SS1 option 30
- Java Developer Kit 1.3, 5722JV1 option 5

Domino for iSeries uses the Java Virtual Machine that ships with OS/400.

**Setup required to run Java programs**

Before running the Java program:

- Set up the PATH environment variable
- Set the CLASSPATH environment variable to include:
  - The file /QIBM/ProdData/Lotus/Notes/Notes.jar
  - The directory that contains the class files for the Java program
- Add QNOTES to the library list:
  
  ```
  ADDLIBLE QNOTES
  ```

  When you run the Java program, submit it under the QNOTES user profile. For example, to run the Java program MyProg, enter this OS/400 command:

  ```
  sbmjob cmd(java class(MyPgm)) user(qnotes) cpyenvvar(*yes)
  ```

  **Note**  For some users or applications, such as WebSphere, it may be easier to set a JVM property, java.library.path=/qsys.lib/qnotes.lib, rather than add QNOTES to the library list.

  For more information, see "About the QNOTES user profile" in Chapter 12 of the *Installing and Managing Domino 6 for iSeries* book (i400help.pdf).

**Using Java in Domino Agents on iSeries**

When you use Java in a Domino agent on iSeries, consider the following tips to help improve the performance of your Domino application:

- Do not put .jar and .zip files into your Domino agents. Instead, put them in stream files in the iSeries file system. Make sure that the QNOTES user profile has read authority to the files. To allow your agents to reference classes in .jar files in the file
system, you'll need to add the .jar file's path to the JavaUserClasses variable in the Domino server's notes.ini file. The JavaUserClasses value becomes part of the Java Virtual Machine's CLASSPATH. If you need more than one path in the JavaUserClasses variable, make sure to separate the values with colons rather than semicolons. For example:

    JavaUserClasses=/QIBM/ProdData/HTTP/public/jt400/lib/jt400.jar:
                      /QIBM/ProdData/Lotus/Notes/NCSO.jar

It is not necessary to put Notes.jar in the JavaUserClasses variable. Ensure that CRTJVAPGM has been run for each of the .jar files. On iSeries, a .class or .jar file will load and execute much faster if a Java program has been created for it. You can use DSPJVAPGM to determine whether a Java program has been created for the .jar files, and what the optimization level of the program is. For best performance, the Java programs should be created with optimization of at least 30. If the Java program has optimization *INTERPRET or less than 30, you should delete it and create one with optimization 30. For example:

    DSPJVAPGM
    CLSF('/qibm/proddata/http/public/jt400/lib/jt400.jar')
    DLTJVAPGM
    CLSF('/qibm/proddata/http/public/jt400/lib/jt400.jar')
    CRTJVAPGM
    CLSF('/qibm/proddata/http/public/jt400/lib/jt400.jar')
    OPTIMIZE(30)

**Note** Running CRTJVAPGM may require heavy use of CPU resources, consider running it in a batch job.

You should check whether a Java program has been created for Notes.jar.
Restrictions and requirements for running Java agents

When you use Java with your Domino for iSeries applications, consider the following:

- When you install the Java Developer Kit on your iSeries server, the Domino Agent Manager (AMgr) and HTTP server automatically support running Java agents. To disable Java agents, add the following line to the NOTES.INI file for the Domino server:

  EnableJavaAgents=0

- Java programs cannot run Java agents or any agents that run Java agents.
- C++ programs cannot run Java agents or any agents that run Java agents.
- The JVM uses file descriptors 0, 1 and 2 for the stdin(), stdout(), and stderr() functions.
- Java Agents on the iSeries cannot load or run native methods that call back into Domino.

Requirements for compiling Java programs, agents, and servlets

You can compile your Java programs, agents, and servlets on any platform that has the Java 1.3 compiler. For the compiler to work properly, the Notes.jar and jsdk.jar files must be specified in the CLASSPATH environment variable. These files are in the following OS/400 directory:

/QIBM/ProdData/Lotus/Notes

For more information about Java and Domino, see the Domino Application development documentation that ships with the Domino Designer application or the Lotus Developer Domain at:

http://www.lotus.com/ldd

JavaMaxHeapSize *NOMAX

There are two new Notes.INI variables:

JavaMaxHeapSize
JavaMinHeapSize

When running Java, these variables allow you to set the level of memory that is available to the JVM to run servlets, agents, or other Java applications.

For Domino's JVM, JavaMaxHeapSize defaults to 2GB and the JavaMinHeapSize defaults to 64 MB. To change these values, enter the number in bytes.

For example, to set maximum heap size to 100 MB (take 100*1024*1024) set JavaMaxHeapSize=104857600.
The same method applies to setting JavaMinHeapSize.

To set the value of JavaMaxHeapSize to *NOMAX, set JavaMaxHeapSize=1 in the INI file. This will allow the JVM to use as much memory as needed.

If the Domino server is using Java extensively, it may be advisable to set the JavaMaxHeapSize to *NOMAX.

---

**JavaOS400CHKPATH**

CHKPATH is a variable that can be set when running any OS/400 Java application, Domino servlets and Domino agents. When using RUNJAVA on iSeries, the CHKPATH option is the equivalent of JavaOS400CHKPATH, a Domino INI variable. CHKPATH means Classpath Security Check Level, which specifies the level of warnings given for directories in the CLASSPATH that have public write authority. A directory in the CLASSPATH that has public write authority is a security exposure because it may contain a class file with the same name as the one you want to run. Whichever class file is found first is run.

Certain Domino tasks create a JVM, such as Amgr and HTTP. In Domino, when the JVM starts, a Java property (os400.class.path.security.check) is set based on the value in JavaOS400CHKPATH. If JavaOS400CHKPATH is not in the INI file, the value of the property defaults to 20, or WARN. CHKPATH must be set to one of the values below or the JVM will fail to start.

The possible values are:

- **JavaOS400CHKPATH=10**
  - IGNORE:
    - Ignores that directories in the CLASSPATH may have public write authority. No warnings messages are sent.

- **JavaOS400CHKPATH=20**
  - WARN:
    - A warning message is sent for each directory in the CLASSPATH that has public write authority. **This is the default setting.**

- **JavaOS400CHKPATH=30**
  - SECURE:
    - A warning message is sent for each directory in the CLASSPATH that has public write authority. If one or more warning messages are sent, an escape message is sent and the Java program is not run.

If JavaOS400CHKPATH is set to WARN(20) or SECURE(30), warnings will appear in the joblog of the task using the JVM. For example, if the directory
"/qibm/proddata/mqm/java/lib" is in the CLASSPATH, has public write authority, and JavaOS400CHKPATH is set to WARN(20) or SECURE(30), the following warning message will be sent to the joblog:

```
Message . . . . : Public write authority on ":<
ddata/mqm/java/lib".

Cause . . . . : Directory ":/qibm/proddata/mqm/java/lib"
in the classpath has public write authority. If more than
one class file in a directory has the same name, the first
one found is run.

Recovery . . . : If the classpath security check level
(CHKPATH) is *SECURE, the command cannot complete if a
directory in the classpath has public write authority. Have
your administrator remove public write authority from
directory ":/qibm/proddata/mqm/java/lib", or change CHKPATH
and run the command again.

If directory ":/qibm/proddata/mqm/java/lib" remains public
write authority

and CHKPATH is not *SECURE, you may verify which classes
are run by setting the OPTION parameter to *VERBOSE and
running the command again.
```

To remove these warnings, set the JavaOS400CHKPATH variable to 10, or
IGNORE, in the INI file and restart the server task for this job log.

---

**Considerations for LotusScript functions on iSeries**

Various LotusScript functions are not supported or work differently because of the
integrated storage management and operating system (OS/400) support.

**Language construct differences**

- **ActivateApp**
  - Not supported. Generates a run-time error.

- **ChDir**
  - A run-time error is generated if LotusScript cannot interpret the argument to
    ChDir; for example, if a drive letter is specified in the argument.
• ChDrive
  Generates a run-time error unless the drive argument is an empty string (""), signifying the default drive.

• CreateObject
  Not supported. Generates a run-time error.

• CurDir, CurDir$
  Generates a run-time error unless the drive argument is the empty string (""), signifying the default drive.

• CurDrive, CurDrive$
  Returns the empty string (""), because there are no drives on iSeries.

• Date, Date$
  Changing the date on iSeries through LotusScript is not supported. Generates a run-time error.

• Declare
  The Pascal calling convention for external function calls is not supported. All external function calls must use the CDECL calling convention. In addition, you must use the _System linkage keyword when passing arguments other than pointers.

• Dir, Dir$
  Ignores the optional attributeMask argument. These functions behave as if all files have the attribute Normal. Returns all files for "*.*", not just those containing ".". Returns those files ending with a period for "*.", not every file without an extension.

• FileLen, Len, LenB, LenBP, LOF
  Strings containing line terminators are smaller than on DOS/Windows platforms. The line terminator is one character (line feed), not two. Therefore, the return value of these functions will be smaller for strings on iSeries than on Windows.

• GetFileAttr
  Generates a run-time error if a drive letter is included in the argument. Does not return the following attributes: ATTR_HIDDEN, ATTR_ARCHIVE, ATTR_VOLUME, ATTR_SYSTEM.

• GetObject
  Not supported. Generates a run-time error.
• Input #, Input, Input$, InputB, InputB$, Line Input, Print, Write #
  Compiled scripts using these constructs may be platform-specific, because file
data is stored in a platform-specific manner. OS/400 character set, byte order,
line terminator, and numeric precision specifics may affect the portability of
scripts using these functions.
• IsObject, IsUnknown
  See "Other differences."
• Open, Lock, Unlock
  Explicit or implicit file locking is not supported. This implies the following:
  • LotusScript for iSeries allows the user to do operations (such as copy or
    open) on a file that is already opened for reading. Thus, the Name statement
    works differently on iSeries.
  • The Open statement can specify only Shared as its lock status. Lock Read,
    Lock Write, and Lock Read Write will cause a run-time error.
  • The Lock and Unlock statements will cause a run-time error.
• SendKeys
  Not supported. Generates a run-time error.
• SetFileAttr
  Ignores the attributes ATTR_HIDDEN, ATTR_ARCHIVE, and
  ATTR_VOLUME.
• Shell
  Window styles are ignored.
• Time, Time$
  Changing the time on iSeries through LotusScript is not supported. Generates a
  run-time error.

File system differences
• There are no drive letters on iSeries. If you use a path name containing a drive
  letter, LotusScript may return an error.
• OS/400 uses the slash (/) character as the directory separator, while
  DOS/Windows use the backslash (\) character. LotusScript supports use of both
  the slash and backslash, with the following restrictions:
  • A Script compiled on any platform other than iSeries or UNIX that uses a
    backslash in a path name string literal will not work on the iSeries.
  • LotusScript cannot use file names (in contrast to path names) that contain the
    backslash character, because this character is always a path separator on
    other platforms.
Other differences

- Function aliasing with ordinal numbers (using the Alias classes in the Declare statement) is not possible on iSeries.

- Where wild cards are permitted in file path strings, LotusScript supports the use of UNIX regular expressions in addition to the "*" and "?" characters. However, using regular expressions in file path strings makes the script platform-dependent.

- OLE is not supported on LotusScript Release 3.1 for iSeries. This difference affects the CreateObject, GetObject, IsObject, and IsUnknown functions. The CreateObject and IsObject functions will raise run-time errors when executed on iSeries platforms. The IsObject function can determine if a variable refers to a native or product object, but not an OLE object, because OLE objects do not exist on the iSeries. The IsUnknown function always returns FALSE on iSeries, because there is no way for a Variant expression to receive the V_UNKNOWN value.

- When passing pointer arguments to C functions, be aware that the pointer size on iSeries is 16 bytes, not 4 bytes.

- All LotusScript statements that write to a file (such as Print #, Put, and Write #) will convert the characters to the platform-specific code page. The iSeries platform uses EBCDIC code pages, which are different from the ASCII code pages used by UNIX or Windows platforms. Therefore, the LotusScript statements will convert the characters to EBCDIC when writing to OS/400 files. Similarly, LotusScript statements that read from a file (such as Get, Input #, and Line Input #) will read OS/400 files as EBCDIC characters.

You can convert the EBCDIC file to ASCII by using the OS/400 Copy Object (CPY) command in your LotusScript program. For example:

Shell("DEL OBJLNK('/acme1/notes/asciifile1.txt')")
Shell("CPY OBJ('/acme1/notes/ebcdicfile.txt')
TOOBJ('/acme1/notes/asciifile1.txt') FROMCODEPAGE(37)
TOCODEPAGE(819)")"
Ways of calling programs from Domino agents

There are several ways for Domino agents to call programs running on the iSeries.

- You can call into C functions within service programs by using the LotusScript "Declare Function" statement. The routines within the service program that you call can call other programs. However, note that the program you are calling is running in a thread-enabled job. Furthermore, if the agent is being initiated through the Web server, the program is running within a thread. There are restrictions on the system regarding what can run in thread-enabled and threaded processes. For example, you cannot call RPG programs from threaded jobs. RPG is not thread-safe. If you can handle these issues, calling C routines gives you the best support for parameter-passing currently. Here is a LotusScript statement that shows how to declare a the routine "runthis" that is in the CmdShell service program (*SRVPGM):

```
Declare Function runthis Lib 
"/qsys.lib/mylib.lib/CmdShell.srvpgm" (Byval cmdstr As String) As Integer
```

The runthis routine takes one parameter, cmdstr. For more information, see the Notes Help database, the Index view, "External declarations".

To call the runthis routine in your LotusScript code, you simply code runthis("parameters_go_here") where "parameters_go_here" are your parameters.

```
cmdstring="call mylibr/getjobinf"
rc=runthis(cmdstring)
Print "Cmdstring: ";cmdstring
```

The C program would look like the following:

```
#include <stdlib.h>

/* This simple program changes the string passed in on cmd to the text 'done calling' */

int runthis(char *cmd)
{
    int rval;
    strcpy(cmd,"done calling");
    return rval;
}
```

As you can see, parameters can be input and output. The string type is unique in that it is passed as a pointer. Make sure you read the related topics under the topic "Calling external C language functions in LotusScript" in Notes Help.
For parameters other than pointers, you should indicate _System linkage in your routine. This is only supported by the iSeries ILE C compiler. Here is an example of the _System linkage statement:

```c
int _System runthis(char *cmd) {
  ...
  /* your routine code goes here */
}
```

The syntax is:

```c
status = odbcResultSet.ExecProcedure(procedureName$, [,arg1] [,arg2] ... [,arg30])
```

or

```c
status = odbcResultSet.ExecProcedure(procedureName$, DB_PARAM_ARRAY, argArray)
```

The parameters are:

- **procedureName$**: String. The name of the procedure you want to execute.
- **arg1...30**: You can pass up to 30 arguments to a procedure. The arguments can be in any format. An argument can serve as input, output, or both. Argument data types must be consistent with the requirements of the procedure. All arguments are separated by commas. Any missing arguments are treated as NULL values. The 30-argument limit is a LotusScript limitation.

To enter over 30 arguments, use the alternative form. The second argument must contain the constant DB_PARAM_ARRAY. The third argument can be an array of any size or type.

The procedure can return values in several forms, depending on the definition of the stored procedure in the relational database system:

- The procedure can return output arguments.
- The procedure can return a result set.
- The procedure can return an execution status, as returned by the back end relational database system.
The Java toolkit for iSeries also has an interface to call programs. If you are using Java in your agent, see this Web site:

http://www.ibm.com/eserver/iseries/java

---

**About adding hook drivers**

On iSeries, you must identify hook drivers by using the prefix "LIB." The name of both the service program and its symbolic link must begin with this prefix. For example, create the service program as LIBHOOK in the library named MYLIB. Then create the symbolic link named LIBHOOK.SRVPGM:

```
addlnk obj('qibm/userdata/lotus/notes/libhook.srvpgm')
newlink('qsys.lib/mylib.lib/libhook.srvpgm')
```

In the NOTES.INI file, specify the NSF_HOOKS statement without the prefix. For example:

```
NSF_HOOKS=HOOK
```
Chapter 3
Integrating Notes and DB2/UDB for iSeries Data

An important feature of Domino for iSeries is the integration between Domino and DB2/UDB for iSeries databases. You can use the following methods to access the DB2 Universal Database for iSeries relational database from Domino applications:

- LotusScript applications use the LotusScript data object (LS:DO) as an interface to DB2/UDB for iSeries data. Through LS:DO, the Domino application sends a request to the DB2/UDB for iSeries database. On other server platforms, LS:DO uses an ODBC interface to relational databases. On iSeries, the LS:DO code passes the request directly to DB2 UDB for iSeries databases without using ODBC. Therefore, the LotusScript program looks the same as it does on other platforms, but the underlying processing on iSeries is more direct.

- Domino formula applications can use @Db functions to access a relational database, including DB2/UDB for iSeries.

- Domino Enterprise Connection Services (DECS) provides real-time access to DB2/UDB for iSeries through a Domino application. Using DECS, you can build live links from Domino pages and forms to data in DB2/UDB for iSeries.

- Lotus Enterprise IntegratorTM provides easy-to-use methods for synchronizing information in Domino databases with information in DB2/UDB for iSeries databases.

Both the LS:DO and the @Db functions are included as part of the Lotus Notes base support.

Be aware of the following requirements and differences when you access data in DB2/UDB for iSeries.

- Security requirements
- Setup to access DB2/UDB for iSeries
- General requirements and differences
- Limit on number of concurrent SQL statements
- Precaution for prestarted jobs that process SQL requests
- LS:DO differences
Remote connection differences
Where to find error messages

DB2/UDB for iSeries provides standards-based SQL run-time support. For details on the SQL syntax for accessing DB2/UDB for iSeries, see the book *IBM SQL Reference* in the Database topic of the iSeries Information Center.

**Setup required to access DB2/UDB for iSeries**

If you use LS:DO or @Db functions to access the DB2/UDB for iSeries relational database, you identify the database by providing its relational database name. For example:

- For LS:DO, specify the relational database name as the data source name in the ConnectTo method.
- For @Db functions, specify the relational database name as the data_source in @DbColumn, @DbCommand, and @DbLookup.

The relational database name is defined by a local relational database entry in the OS/400 relational database directory. You can only have one local relational database entry. Use the Work with Relational Database Directory Entry (WRKRDBDIRE) command to determine if a local relational database entry already exists and add an entry if it does not exist.

**Determining if a local relational database entry exists**

1. On an OS/400 command line, enter:
   
   `wrkrdbdire`

2. In the Remote Location column, look for an entry named *LOCAL.
   - If a *LOCAL entry exists, look for the name under the Relational Database column on the same line. This name is the relational database name.
   - If a *LOCAL entry does not exist, add one.

**Adding a local relational database entry**

1. On an OS/400 command line, enter:
   
   `wrkrdbdire`

2. Type a 1 in the Option field and the name of the relational database in the Relational Database field and press ENTER.
3. Type *LOCAL in the field identified as Remote location: Name or address and press ENTER.
4. Use the default values in the additional fields and press ENTER to add the entry.
About authority when Domino applications access DB2/UDB for iSeries

The methods for accessing DB2/UDB for iSeries from Domino establish a connection from Domino to OS/400. The connection specifies both the user profile whose authority the system uses to access DB2/UDB for iSeries database files and a password for that user profile.

Details: Authority when Domino applications use LS:DO or @Db to access DB2/UDB for iSeries

A Domino application can use either LS:DO (LotusScript data object) or @Db functions to provide access to DB2/UDB for iSeries databases. With both methods, the application establishes a connection with the DB2/UDB for iSeries database. The connection specifies an OS/400 user profile and password. Before allowing the connection, OS/400 checks for the following:

- A valid user profile and password combination.
- The user's authority to the DB2/UDB for iSeries database file.

The following are security considerations for protecting your DB2/UDB for iSeries databases when you provide access from Domino applications.

1. For real-time applications (applications connected to a client), decide which OS/400 user profile the Domino applications will use to access DB2/UDB for iSeries data. You might decide based either on the Domino application or on the DB2/UDB for iSeries database. The following options are available:

   - Use the user profile of the user who is running the Domino application. With this method, you need an OS/400 user profile for every Domino user who needs to run an application that accesses DB2/UDB for iSeries data. "Connecting a Domino application to DB2/UDB for iSeries with a matching OS/400 user profile" describes how your Domino application can provide the OS/400 user profile and password.

   - Set up special OS/400 user profiles whose only function is to provide Domino access to OS/400 data. This eliminates the need for each Domino user to have an OS/400 user profile. "Connecting a Domino application to DB2/UDB for iSeries with a special OS/400 user profile" discusses considerations for this method.

   - Use a combination of these methods. Create special user profiles to provide the equivalent of public (or anonymous) access to Domino users. This technique might be appropriate for database files that every user can view. Rely on the Domino user's OS/400 user profile either for higher levels of access or for confidential files.
2. For scheduled applications (such as agents), you also need to provide an OS/400 user profile when you connect to DB2/UDB for iSeries Scheduled applications run on the server without a connected client. Therefore, the application cannot request a user ID and password from a Domino user. Review "Connecting a Domino application to DB2/UDB for iSeries with a special OS/400 user profile" for alternatives.

Consider using adopted authority to provide tighter control over the actions a Domino program can perform on DB2/UDB for iSeries data.

Connecting a Domino application to DB2/UDB for iSeries with a matching OS/400 user profile
When a Domino application accesses a DB2/UDB for iSeries database, the Domino application needs to establish a connection with DB2/UDB for iSeries. The connection requires a valid OS/400 user profile and password.

When you want your Domino application to connect by using the Domino user's OS/400 profile, do one of the following:

- **Prompt the user**: Your application can prompt the Domino user for an OS/400 user profile name and password during the first connection within a session. Be sure that your application protects this information carefully. You should avoid storing the passwords for individual OS/400 user profiles on your server.

- **Store the user ID and password**: You can provide a form and database for your Domino users who need OS/400 database access. The form would prompt the user for the OS/400 user profile name and password. The application would encrypt the information and store it in a secure database on the client. Therefore, only the user or an application running on that user's behalf would be able to decrypt the password.

When you use this method, your users do not need to enter their OS/400 user profile and password every time they make a connection from Domino to DB2/UDB for iSeries. They will, however, need to use the form to update their database record when they change their OS/400 password.
Connecting a Domino application to DB2/UDB for iSeries with a special OS/400 user profile

When a Domino application accesses a DB2/UDB for iSeries database, the Domino application needs to establish a connection with DB2/UDB for iSeries. The connection requires a valid OS/400 user profile and password.

You might want to create special user profiles for the purpose of providing connections between Domino applications and DB2/UDB for iSeries databases. Do the following for your special user profiles:

1. Decide how many special user profiles to create. Possible options are:
   - A single OS/400 user profile to provide anonymous (or public) access to non-confidential databases.
   - Multiple special-purpose profiles to provide access to DB2/UDB for iSeries data. You might think of these user profiles as similar to group profiles. Their role is to simplify the management of authority. Keep in mind that with this method, OS/400 does not know anything about the real Domino user. The Domino application sets the user profile name. You are relying on the Domino administrator to control who can use the application.

2. Decide whether to use passwords for the connection. Possible options are:
   - Your connection can specify a user profile name and *NOPWD. The QNOTES user profile must have *USE authority to the user profile. With this method, any Domino application can use this OS/400 user profile to attempt to access data.
   - Your application can store the user profile name and a password. You can protect this information so that only trusted programmers can view and update it. However, you must update the application whenever the OS/400 password changes.

   With this method, only Domino users who have authority to the program that contains the user profile and password can attempt to access DB2/UDB for iSeries data with it. The QNOTES profile does not need *USE authority to the user profile.

3. Set up the user profile to protect it from unintended use:
   - Set the initial program to *NONE.
   - Set the initial menu to *SIGNOFF.
Examples: Using adopted authority for Domino access to DB2/UDB for iSeries data

On your iSeries, you might use adopted authority to manage how users update information. For example, the typical user might have *USE authority to the open order file (which allows viewing but not creating, changing, or deleting). However, you want to make sure that only certain users can create or change orders and that a new order passes edit checks before it goes into the open order file. You accomplish this kind of control with adopted authority. A user profile with *CHANGE authority to the open order file owns the program that provides the create and change function. Certain users have *USE authority to run the program.

To use a similar technique when you want to manage the ability to update DB2/UDB for iSeries data from Domino applications, do the following:

1. If necessary, design and create OS/400 programs that perform the desired tasks (such as changing a specific record in a database). You can probably adapt programs that you already have.

2. To set up the programs to adopt the authority of a user profile that has appropriate authority to the database file, do the following:
   - To transfer ownership of the program to the appropriate user profile, use the Change Object Owner (CHGOBJOWN) command.
   - To specify that the program should adopt authority, use the Change Program (CHGPGM) command. Specify *OWNER for the User Profile (USRPRF) parameter.

3. Define the programs as stored procedures for the DB2/UDB for iSeries database files that you want to update.

4. Design and create your Domino programs to use the stored procedures to update the DB2/UDB for iSeries database files.

5. When your Domino programs connect to DB2/UDB for iSeries, specify a user profile name that has *USE authority to the stored-procedure programs.
General requirements and differences for LS:DO and @Db functions

When you use LS:DO or @Db functions to access iSeries data, be aware of the following:

- For LS:DO, you must specify the following "UseLSX" statement in the Event (Options) within LotusScript:
  
  ```LotusScript
  UseLSX "*LSXODBC"
  ```

  This requirement is similar to other platforms.
- For @Db functions, the first parameter must be "ODBC".
- When you specify SQL statements for the LotusScript ODBCQuery class or in @Db functions, make sure you use the SQL naming convention. That is, use:
  
  ```LotusScript
  collection.table
  ```

  not:

  ```LotusScript
  collection/table
  ```

- For iSeries, the LS:DO and @Db functions require that you specify a user ID and password. However, there is no interactive prompting for the user ID and password and no support for automatic registration. Therefore, you must specify the user ID and password in your script.
- For the best performance on updates, deletes, and inserts, specify the SQL UPDATE, DELETE, and INSERT statement in the odbcqry.SQL property and then use the ODBCResult.Execute method.
- The LS:DO and @Db functions use the SQL Call Level Interface (CLI) instead of ODBC. However, the user interface is the same.

  The CLI does not have an ODBC.INI file. You must register the data source on OS/400 by using the Work with Relational Database Entries (WRKRDBDIRE) command to add the data source to the OS/400 relational database directory.

  The SQL CLI is described in the iSeries book *DB2 Universal Database for iSeries SQL Call Level Interface*, available in PDF format from the following Web address:

  http://www.ibm.com/eserver/iseries/infocenter
Limit on number of concurrent SQL statements

DB2/UDB for iSeries limits the number of internal handles that can be used for processing SQL requests. This limit may cause problems with the @Db or LS:DO functions run by the Domino HTTP server.

These problems are indicated by a message in the Domino HTTP server job log, such as:

*Error Occurred in SQL Call Level Interface. Reason code of 14.*

If you see such a message, change the number of threads that the HTTP server can use to process requests. You can change this number by changing the settings in the Server document in the Domino Directory for the Domino server. In the Number of active threads field, specify a number of threads that is no more than 100.

Precaution for prestarted jobs that process SQL requests

DB2/UDB for iSeries uses OS/400 prestarted jobs for processing SQL requests. These jobs are started when the Domino server is started on iSeries through the STRDOMSVR command. If the Domino server is not started when you run standalone applications that call @Db or LS:DO agents, you may need to explicitly start the prestarted jobs. You need to start the prestarted jobs if you see a message such as the following in the Agent Manager job log:

*No authority has been granted to use the command*

To start the prestarted jobs, enter this OS/400 command:

```
strpj sbs(qsyswrk) pgm(qsys/qsqsrvr)
```

You could also encounter problems if you have altered the subsystem description for the QSYSWRK subsystem and removed the entry for the QSQRVR prestart job.

Differences for LS:DO

When you use LS:DO to access DB2/UDB for iSeries, be aware of the following differences.

**ODBCConnection class**

- ListTables method
  
  This method returns a list of all the tables in all the libraries. Therefore, the result set can be so large that it cannot be returned.

- ListFields method
  
  Do not specify the optional datasource parameter. For OS/400, a connection must already be established before calling ListFields.
• ListDataSource method
  If you specify the datasource parameter, you must also specify a user ID and password to establish a connection. Otherwise, the current connection is used.

• ListProcedures method
  If you specify the datasource parameter, you must also specify a user ID and password. If the data source is not specified, the connection that is currently established is used.

ODBCQuery class
• If the tables from the datasource contain a large number of columns (greater than 100), avoid using "*" in the SQL statement to list all columns. Instead, specify the actual column names that you want to retrieve. For example, the following SQL statement might result in a "memory error" if the table contains more than 100 columns:

  ```sql
  Myquery.SQL = "select * from lib.table"
  ```

  Instead, use a SQL statement such as the following:

  ```sql
  Myquery.SQL = "select col1, col2, col3 from mylib.mytable"
  ```

ODBCResult class
• Do not use quotation marks (double quotes) around the library name, table name, or column name. For example, the following command will not work:

  ```csharp
  Result1.DeleteRow("mylib.""mytable"")
  ```

• When setting DateTime values using the SetValue method, you must enclose the values in quotation marks as strings. The values must be in the following format:

<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>&quot;yyyy-mm-dd&quot;</td>
<td>&quot;1997-07-31&quot;</td>
</tr>
<tr>
<td>Time</td>
<td>&quot;hh:mm:ss&quot;</td>
<td>&quot;12:15:30&quot;</td>
</tr>
<tr>
<td>Time Stamp</td>
<td>&quot;yyyy-mm-dd-hh.mm.ss.mmm&quot;</td>
<td>&quot;1997-07-31-12.15.35.00000&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;mmm&quot;</td>
<td>&quot;000&quot;</td>
</tr>
</tbody>
</table>

• If the query statement specified on the SQL property of the ODBCQuery class contains a column with REAL (single-precision floating-point) data type, the UpdateRow method might not work. The error #546 is returned:

  LS:DO- The result contains no data.
• If the query statement contains multiple columns of the same name from different tables and the column names are qualified with the alias table names, the UpdateRow method does not work. To avoid this problem, qualify the column names with the full "library.table" names. Here are examples.

This SQL statement does not work:
```
Select t1.cusnum, t2.cusnum, t2.balance from mylib.mytable1 t1, mylib.mytable2 t2 where t1.cusnum=t2.cusnum
```

This SQL statement works:
```
Select mylib.mytable1.cusnum, mylib.mytable2.cusnum, mylib.mytable2.balance from mylib.mytable1, mylib.mytable2 where mylib.mytable1.cusnum=mylib.mytable2.cusnum
```

• Use the Close method to close a result set and free the associated resource before executing another SQL statement; for example:
```
Result.Close(DB_CLOSE)
```

• The FieldInfo method returns an array of elements. The following elements are not supported on iSeries:
  • DB_INFO_UNSIGNED
  • DB_INFO_MONEY
  • DB_INFO_READONLY (always returns 1)
  • DB_INFO_AUTOINCREMENT
  • DB_INFO_CASESENSITIVE
  • DB_INFO_SEARCHABLE
  • DB_INFO_SETTABLE (always returns -1)

The DB_INFO_LENGTH element (same as the ResultSet.FieldSize method) is applicable for specific data types:

<table>
<thead>
<tr>
<th>SQL data type</th>
<th>DB_INFO_LENGTH indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer, small int</td>
<td>number of bytes</td>
</tr>
<tr>
<td>float</td>
<td></td>
</tr>
<tr>
<td>char, varchar,</td>
<td>number of characters</td>
</tr>
<tr>
<td>long varchar</td>
<td></td>
</tr>
<tr>
<td>numeric, decimal</td>
<td>(not applicable)</td>
</tr>
</tbody>
</table>

The DB_INFO_PRECISION and DB_INFO_SCALE elements are applicable only for numeric or decimal data types.
• Use the ExecProcedure method in the ODBCResultSet object to run a stored procedure.

The syntax is:

```
status = odbcResultSet.ExecProcedure(procedureName$ [,arg1] [,arg2] ... [,arg30])
```
or

```
status = odbcResultSet.ExecProcedure(procedureName$,
   DB_PARAM_ARRAY, argArray)
```

The parameters are:

- `procedureName$`  
  String. The name of the procedure you want to execute. For example, library.procedure.

- `arg1...30`  
  You can pass up to 30 arguments to a procedure. The arguments can be in any format. An argument can serve as input, output, or both. Argument data types must be consistent with the requirements of the procedure. All arguments are separated by commas. Any missing arguments are treated as NULL values. The 30-argument limit is a LotusScript limitation.

  To enter over 30 arguments, use the alternative form. The second argument must contain the constant `DB_PARAM_ARRAY`. The third argument can be an array of any size or type.

  The procedure can return values in several forms, depending on the definition of the stored procedure in the relational database system:

  - The procedure can return output arguments.
  - The procedure can return a result set.
  - The procedure can return an execution status, as returned by the back end relational database system.

**Remote connection differences**

When you access data from a remote iSeries for the first time, the request may fail with the following error message:

```
SQL package QSQCLIPKGN in QGPL not found
```

**Note** You retrieve messages by using the GetExtendedErrorMessage method.

If the request fails with this error message, you need to create the required SQL package on the remote iSeries.
First, sign onto the remote iSeries and perform the following operations:

1. Enter the command:
   
   \texttt{wrkobj obj(qgpl/*all) objtype(*sqlpkg)}

2. Look for these two objects:
   - QSQCLIPKGC
   - QSQCLIPKGN

3. If the object QSQCLIPKGC exists, delete it:
   
   \texttt{dltsqlpkg sqlpkg(qgpl/qsqclipkgc)}

4. If the object QSQCLIPKGN exists, delete it:
   
   \texttt{dltsqlpkg sqlpkg(qgpl/qsqclipkgn)}

Next, use the Notes client to create and run the following LS:DO script on the local iSeries or add the script to the beginning of your existing LS:DO script.

\textbf{Event Options:}

- Option Public
- \texttt{Uselsx "*lsxodbc"}

\textbf{Event Initialize:}

\begin{verbatim}
Sub Initialize
  Dim con As New ODBCConnection
  con.AutoCommit=False
  If (con.ConnectTo("DataSource","UserID","Password")) Then
    Call con.Disconnect
  End If
End Sub
\end{verbatim}

To verify that your script ran successfully, repeat steps 1 and 2 above.

After you successfully run your script, delete the script you added. The added script only needs to run once to create the SQL package on the remote system.
Where to find error messages

Some SQL messages are logged in the OS/400 job logs, such as the Agent Manager job log or the HTTP server job log. For the LS:DO function, you can use the GetError, GetErrorMessage, and GetExtendedErrorMessage methods to log any error messages to the server console or to a file.

You can log additional information about SQL calls to the server console by doing the following:

1. From the console, enter:
   
   ```
   SET CONFIG lsxodbc_gdf_level =2
   (or specify a level of 0 to turn off the logging)
   ```

2. End the job that processes the LS:DO or @Db requests. For example, from the console, enter:
   
   ```
   tell AMgr quit
   ```

3. Restart the job. For example, from the console, enter:
   
   ```
   load AMgr
   ```
Chapter 4
Domino for iSeries APIs

This chapter describes APIs unique to Domino for iSeries and usage of the Management Collection Object API.

Domino for iSeries server APIs

There are a number of API functions that provide program access to a Domino server:

- **List Domino Servers (QnninListDominoServers)**: is used to retrieve the list of Domino servers on the system.
- **Retrieve Domino Server Information (QnninRtvDominoServerI)**: is used to retrieve specific information about each Domino server.
- **Retrieve Domino Server Attributes (QnninRtvDominoServerAttr)**: is used to retrieve specific information about each Domino server. This API program retrieves all of the information provided by QnninRtvDominoServerI plus additional information based on the format name.
- **Set Domino Environment (QnninSetDominoEnv)**: is used to set the current jobs working environment into a state to allow the NotesInitExtended API to be called for a specific Domino server. This removes the burden from the caller from having to know this specific information about each Domino server when initializing the Notes API environment.
- **Get Domino Environment (QnninGetDominoEnv)**: is used to retrieve information about the current jobs Domino server environment
- **List Domino Release Information (QnninListDominoRlsI)**: is used to retrieve a list of installed Domino releases on the current iSeries system.
- **Get Notes.Ini Value (QnninGetIniValue) and QnninGetIniValuez**: are used to retrieve a value from a Domino server's notes.ini file.
- **Set Notes.Ini Value (QnninSetIniValue) and QnninSetIniValuez**: is used to set a value in the Domino server's notes.ini file.
- **Get Server Document Item (QnninGetServerDocItem) and QnninGetServerDocItemz**: is used to retrieve an item value from a Domino server's server document found in the Domino directory (names.nsf file).
• **Set Server Document Item (QnninSetServerDocItem) and QnninSetServerDocItemz:** is used to change an item value in a Domino server's server document found in the Domino directory (names.nsf file).

You can download examples for using these APIs at:


These API functions are included in the base option of Domino for iSeries, 5733LD6. If 5722SS1 option 13 (OS/400 System Openness Includes) has already been installed when Domino for iSeries version 6.0.0 or later is installed, then a symbolic link will be created in the IFS to the associated header file that is in the QNOTES library. If you expect developers to need the symbolic link from IFS, then make sure that 5722SS1 option 13 is installed prior to installing Domino for iSeries version 6.0.0 or later. If option 13 is not installed on the system at the time Domino is installed, then you must create a symbolic link to the header file using the command as shown below:

```bash
QSYS/ADDLNK OBJ('/QSYS.LIB/QNOTES.LIB/H.FILE/QNNINLDS.MBR') NEWLNK('/QIBM/INCLUDE/QNNINLDS.H')
```

**Note** The directory '/QIBM/INCLUDE' must exist on the system for this command to work. It is generally created after installing 5722SS1 option 13. The directory should be owned by QSYS and have public *RX authority.

---

**List Domino Servers (QnninListDominoServers)**

**Required Parameter Group:**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data buffer for Domino servers</td>
<td>O</td>
<td>Char(*)</td>
</tr>
<tr>
<td>2</td>
<td>Data buffer length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>3</td>
<td>Format name</td>
<td>I</td>
<td>Char(8)</td>
</tr>
<tr>
<td>4</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

**Service Program Name:** QNNINLDS

**Similar Commands:** None

**Authorities and Locks**

**Default public authority**

*USE

46  Domino 6 for iSeries Application Development Guide
Required Parameter Group
Data buffer for Domino servers

OUTPUT; CHAR(*)

This returns the list of Domino servers in the buffer. For the format, see "Data Buffer." The server names are returned in the CCSID of the job that is currently running.

Data buffer length

INPUT; BINARY(4)

The length of the data buffer. The length must be big enough to hold at least the Bytes returned field and the Bytes available field. Failure to provide enough room for the data will result in errors or incomplete data being returned.

Format name

INPUT; CHARACTER(8)

The name of the format used to retrieve all of the configured Domino servers. You can use the following format name: DSRV0100

Server name

INPUT; CHARACTER(255)

The name of the Domino server to retrieve information. This is a blank padded field. If the server name was 10 characters long, then there would be 245 blanks following the server name.

Server name length

INPUT; BINARY(4)

The length of the server name including the blanks. It should always be 255.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.
Data Buffer

The following information is returned in the input parameter section for format name DSRV0100. For detailed descriptions of the fields in this table, see Field Descriptions.

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Binary(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Binary(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Binary(4)</td>
<td>Offset to server entry</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Binary(4)</td>
<td>Number of server entries returned</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Binary(4)</td>
<td>Length of server entry</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Character</td>
<td>Server entries (*)</td>
</tr>
</tbody>
</table>

Field Descriptions

**Bytes available** - The length of data that could be returned by this API. This value is greater than "Bytes returned" when the length of the receiving variable is too small, causing truncation of data.

**Bytes returned** - The length of data returned in this structure. This value includes this and all following fields. If insufficient space is provided for the receiver value, this value would be set to the last byte of the last complete array entry.

**Length of server entry** - This is the length of each server entry returned. It is 255 bytes.

**Number of server entries returned** - This is the number of server entries that can be found in the returned "Server entries" field. There could be a 0 returned for this field if there isn't enough space to hold the first server entry or no Domino servers are currently configured.

**Offset to server entry** - This is the offset to the start of the server entries. The offset is from the start of the "Bytes returned" field.

**Server entries** - This is the start of the server entries returned. Each server entry is the length specified in the "length of server entry" field.
Error Messages
CPF3CF1: Error code parameter not valid
CPF3CF2: Error(s) occurred during running of &1 API.
CPF3C1E: Required parameter &1 omitted.
CPF3C21 E: Format name &1 is not valid.
CPF3C24 E: Length of the receiver variable is not valid.

Retrieve Domino Server Information (QnninRtvDominoServerI)

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Required Parameter Group</th>
<th>O/O</th>
<th>Char(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data buffer for Domino server information</td>
<td>O</td>
<td>Char(*)</td>
</tr>
<tr>
<td>2</td>
<td>Data buffer length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>3</td>
<td>Server name</td>
<td>I</td>
<td>Char(255)</td>
</tr>
<tr>
<td>2</td>
<td>Server name length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>4</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Service Program Name: QNNINLDS

Similar Commands: None

Authorities and Locks
Default public authority

*USE

Required Parameter Group
Data buffer for Domino servers
OUTPUT; CHAR(*)

This returns the information about a specific Domino server. For the format, see "Data Buffer".

Data buffer length
INPUT; BINARY(4)

The length of the data buffer. The length must be at least big enough to hold the Bytes returned field and Bytes available field. Failure to provide enough room for the data will result in errors or incomplete data being returned.

Chapter 4: Domino for iSeries APIs  49
Server name

INPUT; CHARACTER(255)

The name of the Domino server to retrieve information. This is a blank padded field. If the server name was 10 characters long, then there would be 245 blanks following the server name.

Server name length

INPUT; BINARY(4)

The length of the server name including the blanks. It should always be 255.

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Data Buffer

For detailed descriptions of the fields in this table, see Field Descriptions later in this section

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Binary(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Binary(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Binary(4)</td>
<td>Primary type of Domino server</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Binary(4)</td>
<td>Secondary type of Domino server</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Binary(4)</td>
<td>Number of active jobs in the subsystem</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Binary(4)</td>
<td>Offset to data directory path</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Binary(4)</td>
<td>Length of data directory path</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>Binary(4)</td>
<td>Offset to executable directory path</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>Binary(4)</td>
<td>Length of executable directory path</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>Character(20)</td>
<td>Subsystem description</td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>Character(10)</td>
<td>Library name</td>
</tr>
<tr>
<td>66</td>
<td>42</td>
<td>Binary(1)</td>
<td>Server status</td>
</tr>
<tr>
<td>67</td>
<td>43</td>
<td>Binary(1)</td>
<td>Auto-Start with TCP/IP Servers</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>Binary(4)</td>
<td>Partition Number</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>Binary(4)</td>
<td>Offset to generic information</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>Binary(4)</td>
<td>Length of generic information</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>Character(*)</td>
<td>Variable data</td>
</tr>
</tbody>
</table>
Field Descriptions

Auto - Start with TCP/IP servers - This is a flag that indicates if the server will be started automatically when TCP/IP is started on the server.

0 - The server will not be started with TCP/IP
1 - The server will be started with TCP/IP

Bytes available - The length of data that could be returned by this API. This value is greater than "Bytes returned" when the length of the receiving variable is too small, causing truncation of data.

Bytes returned - The length of data returned in this structure. This value includes this and all following fields. If the data is truncated because the receiver variable is not large enough to hold all of the data available, this value will be less than the bytes available value.

Length of data directory path - This is the length of the data in the data directory path.

Length of executable directory path - This is the length of the data in the executable directory path.

Length of generic information - This is the length of the data in the generic information.

Library name - This is the library name of the run time programs.

Number of active jobs in the subsystem - This is the number of active jobs in the subsystem associated with this Domino server. If the Domino server is currently not running this value will be 0. If a -1 is returned then the number of active jobs running in the subsystem could not be determined, and a diagnostic message is written that gives more information on why this information could not be retrieved.

Offset to data directory path - This is the offset, from the start of the returned data, to where the data directory path can be found. It will be somewhere within the variable data portion of the returned data.

Offset to executable directory path - This is the offset, from the start of the returned data, to where the executable directory path can be found. It will be somewhere within the variable data portion of the returned data.

Offset to generic information - This is the offset, from the start of the returned data, to where the generic configuration information can be found. It will be somewhere within the variable data portion of the returned data.

Partition Number - The internal partition number used by the Domino server to identify itself within a partitioned server environment.
**Primary type of Domino server** - This is the type of Domino server. The possible values are:

- 0 - Unknown Domino server type
- 1 - Domino server
- 2 - Stand alone QuickPlace server
- 3 - Stand alone Sametime server

If the server type is a Domino server, you may need to check the **Secondary type of Domino server** field to see if there are other capabilities of the server.

**Secondary type of Domino server** - This Domino server also has these additional capabilities. The possible values are:

- 0 - No additional capabilities.
- 1 - Also a QuickPlace server
- 2 - Also a Sametime server
- 3 - Also a QuickPlace AND Sametime server

This field is only valid if the **Primary type of Domino server** field indicates a Domino server.

**Server status** - This is the current status of the server. The possible values are:

- 1 - Server ended
- 2 - Server started
- 3 - Server starting
- 4 - Server ending
- 5 - Server in standby mode
- 99 - Server in unknown status

**Subsystem description** - This is the name of the subsystem description used for this Domino server. The first 10 bytes of the data will be the subsystem description name and the next 10 bytes will be the library name where the subsystem description can be found.

**Variable data** - This is the start of the variable data field. Use the offset and length fields to actually find and retrieve data from this area.
Error Messages
CPF24B4: Severe error while addressing parameter list.
CPF3CF1: Error code parameter not valid
CPF3CF2: Error(s) occurred during running of &1 API.
CPF3C1E: Required parameter &1 omitted.
CPF3C24: Length of the receiver variable is not valid.
LNT0907: The server name specified, &1, is not valid.

Retrieve Domino Server Attributes (QnninRtvDominoServerAttr)

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>ParameterDirection</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data buffer for Domino server information</td>
<td>O</td>
<td>Char(*)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Data buffer length</td>
<td>I</td>
<td>Binary(4)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Server name</td>
<td>I</td>
<td>Char(255)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Server name length</td>
<td>I</td>
<td>Binary(4)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Format name</td>
<td>I</td>
<td>Char(8)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Error Code</td>
<td>I/</td>
<td>Char(*)</td>
<td></td>
</tr>
</tbody>
</table>

Service Program Name: QNNINLDS

Similar Commands: None

Authorities and Locks
Default public authority
*USE

Required Parameter Group
Data buffer for Domino server information
Data buffer for Domino server information
OUTPUT; CHAR(*)

This returns the information about a specific Domino server. For the format, see "Data Buffer."
**Data buffer length**

INPUT; BINARY(4)

The length of the data buffer. The length must be at least big enough to hold the Bytes returned field and Bytes available field. Failure to provide enough room for the data will result in errors or incomplete data being returned.

**Server name**

INPUT; CHARACTER(255)

The name of the Domino server to retrieve information. This is a blank padded field. If the server name was 10 characters long, then there would be 245 blanks following the server name.

**Server name length**

INPUT; BINARY(4)

The length of the server name including the blanks. It should always be 255.

**Format name**

INPUT; CHARACTER(8)

The name of the format used to retrieve different Domino server attributes. You can use the following format names:

DATR0100, DATR0200

**Error code**

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

**Data Buffer**

For detailed descriptions of the fields in these tables, see the Field Descriptions later in this section.
### DATR0100 Attribute Buffer

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Binary(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Binary(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Binary(4)</td>
<td>Primary type of Domino server</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Binary(4)</td>
<td>Secondary type of Domino server</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Binary(4)</td>
<td>Number of active jobs in the subsystem</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Binary(4)</td>
<td>Offset to data directory path</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Binary(4)</td>
<td>Length of data directory path</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>Binary(4)</td>
<td>Offset to executable directory path</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>Binary(4)</td>
<td>Length of executable directory path</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>Character(20)</td>
<td>Subsystem description</td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>Character(10)</td>
<td>Library name</td>
</tr>
<tr>
<td>66</td>
<td>42</td>
<td>Binary(1)</td>
<td>Server status</td>
</tr>
<tr>
<td>67</td>
<td>43</td>
<td>Binary(1)</td>
<td>Auto-Start with TCP/IP Servers</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>Binary(4)</td>
<td>Partition Number</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>Binary(4)</td>
<td>Offset to generic information</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>Binary(4)</td>
<td>Length of generic information</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>Character(*)</td>
<td>Variable data</td>
</tr>
</tbody>
</table>

### DATR0200 Attribute Buffer

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>DATR0100 Buffer (see above)</td>
<td>DATR0100 Buffer</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>Character(1)</td>
<td>Log replication events</td>
</tr>
<tr>
<td>81</td>
<td>51</td>
<td>Character(1)</td>
<td>Log client session events</td>
</tr>
<tr>
<td>82</td>
<td>52</td>
<td>Character(1)</td>
<td>DOLS enabled</td>
</tr>
<tr>
<td>83</td>
<td>53</td>
<td>Character(1)</td>
<td>Day light savings time enabled</td>
</tr>
<tr>
<td>84</td>
<td>54</td>
<td>Character(6)</td>
<td>Time zone</td>
</tr>
<tr>
<td>90</td>
<td>5A</td>
<td>Character(10)</td>
<td>Collation sort order</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>Character(11)</td>
<td>SMTP type</td>
</tr>
<tr>
<td>111</td>
<td>6F</td>
<td>Character(1)</td>
<td>Directory type</td>
</tr>
<tr>
<td>112</td>
<td>70</td>
<td>Character(1)</td>
<td>Application Server Provider</td>
</tr>
<tr>
<td>113</td>
<td>71</td>
<td>Character(3)</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Chapter 4: Domino for iSeries APIs  55
<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>116</td>
<td>74</td>
<td>Binary(4)</td>
<td>Offset to organization name</td>
</tr>
<tr>
<td>120</td>
<td>78</td>
<td>Binary(4)</td>
<td>Length of organization name</td>
</tr>
<tr>
<td>124</td>
<td>7C</td>
<td>Binary(4)</td>
<td>Offset to web browser list, List Type 1</td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td>Binary(4)</td>
<td>Offset to News Readers List, List Type 1</td>
</tr>
<tr>
<td>132</td>
<td>84</td>
<td>Binary(4)</td>
<td>Offset to Mail services List, List Type 1</td>
</tr>
<tr>
<td>136</td>
<td>88</td>
<td>Binary(4)</td>
<td>Offset to Advanced services List, List Type 1</td>
</tr>
<tr>
<td>140</td>
<td>8C</td>
<td>Binary(4)</td>
<td>Offset to Connection Services List, List Type 1</td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td>Binary(4)</td>
<td>Offset to Directory Services List, List Type 1</td>
</tr>
<tr>
<td>148</td>
<td>94</td>
<td>Binary(4)</td>
<td>Offset to Additional Services List, List Type 1</td>
</tr>
<tr>
<td>152</td>
<td>98</td>
<td>Binary(4)</td>
<td>Offset to TCP Info List, List Type 2</td>
</tr>
<tr>
<td>156</td>
<td>9C</td>
<td>Binary(4)</td>
<td>Offset to canonical name</td>
</tr>
<tr>
<td>160</td>
<td>A0</td>
<td>Binary(4)</td>
<td>Length of canonical name</td>
</tr>
<tr>
<td>164</td>
<td>A4</td>
<td>Binary(4)</td>
<td>Offset to server title</td>
</tr>
<tr>
<td>168</td>
<td>A8</td>
<td>Binary(4)</td>
<td>Length of server title</td>
</tr>
<tr>
<td>172</td>
<td>AC</td>
<td>Binary(4)</td>
<td>Offset to server host name</td>
</tr>
<tr>
<td>176</td>
<td>B0</td>
<td>Binary(4)</td>
<td>Length of server host name</td>
</tr>
<tr>
<td>180</td>
<td>B4</td>
<td>Binary(4)</td>
<td>Offset to domain name</td>
</tr>
<tr>
<td>184</td>
<td>B8</td>
<td>Binary(4)</td>
<td>Length of domain name</td>
</tr>
<tr>
<td>188</td>
<td>BC</td>
<td>Binary(4)</td>
<td>Offset to administrator(s) name(s)</td>
</tr>
<tr>
<td>192</td>
<td>C0</td>
<td>Binary(4)</td>
<td>Length of administrator(s) name(s)</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>Character(*)</td>
<td>Variable data</td>
</tr>
</tbody>
</table>

Field Descriptions

Application Service Provider - This is a flag that indicates if the server is configured for Application Service Provider (ASP). The possible values are:

'0' - Not configured for ASP

'1' - Configured for ASP
**Auto-Start with TCP/IP servers** - This is a flag that indicates if the server will be started automatically when TCP/IP is started on the system.

0 - The server will not be started with TCP/IP
1 - The server will be started with TCP/IP

**Bytes available** - The length of data that could be returned by this API. This value is greater than "Bytes returned" when the length of the receiving variable is too small, causing truncation of data.

**Bytes returned** - The length of data returned in this structure. This value includes this and all following fields. If the data is truncated because the receiver variable is not large enough to hold all of the data available, this value will be less than the bytes available value.

**Collation sort order** - This field indicates which way the Domino server will sort characters. The possible values are:

<table>
<thead>
<tr>
<th>Collation</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>*SAME</td>
<td>HU - Hungarian</td>
</tr>
<tr>
<td>*STD - Standard</td>
<td>IS - Icelandic</td>
</tr>
<tr>
<td>CS - Czech</td>
<td>IT - Italian</td>
</tr>
<tr>
<td>DA-DK-AA - Nordic 2to1</td>
<td>JA - Japanese</td>
</tr>
<tr>
<td>DE- German</td>
<td>KO - Korean</td>
</tr>
<tr>
<td>E2-ES - Spanish - Modern</td>
<td>LT - Lithuanian</td>
</tr>
<tr>
<td>EL- Greek</td>
<td>LV - Baltic</td>
</tr>
<tr>
<td>EN - Latin1</td>
<td>NL- Dutch</td>
</tr>
<tr>
<td>ES - Spanish - Traditional</td>
<td>NO- Nordic</td>
</tr>
<tr>
<td>ET-EE - Estonian collation</td>
<td>PL - Latin2</td>
</tr>
<tr>
<td>PL-PL - Polish</td>
<td>FR - French</td>
</tr>
<tr>
<td>HR- Croatian</td>
<td>RO - Romanian</td>
</tr>
<tr>
<td>RO - Romanian</td>
<td>RU - Cyrillic</td>
</tr>
<tr>
<td>HR- Croatian</td>
<td>RU - Cyrillic</td>
</tr>
<tr>
<td>RO - Romanian</td>
<td>RU - Cyrillic</td>
</tr>
<tr>
<td>HR- Croatian</td>
<td>RU - Cyrillic</td>
</tr>
</tbody>
</table>

**Directory Type** - This is a flag that indicates if the server is configured as a central directory or a config-only directory. The possible values are:

'1' - Central Directory

'2' - Config-Only Directory

**DOLS enabled** - This is a flag that indicates if the server is enabled for Domino Offline Support (DOLS). The possible values are:

'0' - Not enabled for DOLS

'1' - Enabled for DOLS
DST - This is a flag that indicates if the server is configured for Daylight Savings Time (DST). The possible values are:

'0' - Not configured for DST

'1' - Configured for DST

Length of administrator name - This is the length of the data in the administrator name.

Length of canonical name - This is the length of the data in the canonical name. If length is equal to 0 (zero) then no canonical server name is available.

Length of data directory path - This is the length of the data in the data directory path.

Length of domain name - This is the length of the data in the domain name.

Length of executable directory path - This is the length of the data in the executable directory path.

Length of generic information - This is the length of the data in the generic information.

Length of organization name - This is the length of the data in the organization name.

Length of server host name - This is the length of the data in the server host name. If length is equal to 0 (zero) then no server host name is available.

Length of server title - This is the length of the data in the server title. If length is equal to 0 (zero) then no server title is available.

Library name - This is the library name of the run time programs

Log client session events - This is a flag that indicates if client session events should be logged. The possible values are:

'0' - Do not log client session events

'1' - Log client session events.

Log replication events - This is a flag that indicates if replication events should be logged. The possible values are:

'0' - Do not log replication events

'1' - Log replication events.

Number of active jobs in the subsystem - This is the number of active jobs in the subsystem associated with this Domino server. If the Domino server is currently not running this value will be 0. If a -1 is returned then the number of active jobs running in the subsystem could not be determined. If a -1 is returned there is a diagnostic message written that gives some more information on why this information could not be retrieved.
Offset to administrator name - This is the offset, from the start of the returned data, to where the administrator name can be found. It will be somewhere within the variable data portion of the returned data.

Offset to advanced services list - This is the offset, from the start of the returned data, to the list of advanced services of the server. This list is in List Type 1 format. See below for the description of the List Type 1 format.

Offset to canonical name - This is the offset, from the start of the returned data, to where the canonical name can be found. It will be somewhere within the variable data portion of the returned data. Example of canonical server name:

    CN=MYSERVER/OU=SALES/O=ORG1/C=US

Offset to connection services list - This is the offset, from the start of the returned data, to the list of connection services of the server. This list is in List Type 1 format. See below for the description of the List Type 1 format.

Offset to data directory path - This is the offset, from the start of the returned data, to where the data directory path can be found. It will be somewhere within the variable data portion of the returned data.

Offset to directory services list - This is the offset, from the start of the returned data, to the list of directory services of the server. This list is in List Type 1 format. See below for the description of the List Type 1 format.

Offset to domain name - This is the offset, from the start of the returned data, to where the domain name can be found. It will be somewhere within the variable data portion of the returned data.

Offset to executable directory path - This is the offset, from the start of the returned data, to where the executable directory path can be found. It will be somewhere within the variable data portion of the returned data.

Offset to generic information - This is the offset, from the start of the returned data, to where the generic configuration information can be found. It will be somewhere within the variable data portion of the returned data.

Offset to mail services list - This is the offset, from the start of the returned data, to the list of internet mail packages of the server. This list is in List Type 1 format. See below for the description of the List Type 1 format.

Offset to news readers list - This is the offset, from the start of the returned data, to the list of news readers of the server. This list is in List Type 1 format. See below for the description of the List Type 1 format.

Offset to organization name - This is the offset, from the start of the returned data, to where the organization name can be found. It will be somewhere within the variable data portion of the returned data.
Offset to server host name - This is the offset, from the start of the returned data, to where the server host name can be found. It will be somewhere within the variable data portion of the returned data.

Offset to server title - This is the offset, from the start of the returned data, to where the server title can be found. It will be somewhere within the variable data portion of the returned data.

Offset to TCP/IP information list - This is the offset, from the start of the returned data, to the list of TCP/IP port information of the server. This list is in List Type 2 format. See below for the description of the List Type 2 format.

Offset to web browsers list - This is the offset, from the start of the returned data, to the list of web browser features of the server. This list is in List Type 1 format. See below for the description of the List Type 1 format.

Partition Number: The internal partition number used by the Domino server to identify itself within a partitioned server environment.

Primary type of Domino server: This is the type of Domino server. The possible values are:

0 - Unknown Domino server type
1 - Domino server
2 - Stand alone QuickPlace server
3 - Stand alone Sametime server

If the server type is a Domino server, you may need to check the Secondary type of Domino server field to see if there are other capabilities of the server.

Secondary type of Domino server - This Domino server also has these additional capabilities. The possible values are:

0 - No additional capabilities.
1 - Also a QuickPlace server
2 - Also a Sametime server
3 - Also a QuickPlace AND Sametime server

This field is only valid if the Primary type of Domino server field indicates a Domino server.
**Server status** - This is the current status of the server. The possible values are:

1 - Server ended
2 - Server started
3 - Server starting
4 - Server ending
5 - Server in standby mode
99 - Server in unknown status

**SMTP Type** - This field indicates how the Domino server supports SMTP mail. The possible values are:

*MSF*  
AS/400 SMTP server in combination with the AnyMail/400 Mail Server Framework

*DOMINO*  
Built-in Domino SMTP

*QUICKPLACE*  
Built-in QuickPlace SMTP (QuickPlace servers only)

**Subsystem description** - This is the name of the subsystem description used for this Domino server. The first 10 bytes of the data will be the subsystem description name and the next 10 bytes will be the library name where the subsystem description can be found.
**Time Zone** - This field indicates what time zone the server is configured for. The possible values are:

<table>
<thead>
<tr>
<th>Time Zone</th>
<th>Offset Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMT - Greenwich Mean Time</td>
<td>BST - Bering Standard Time</td>
</tr>
<tr>
<td>ZW1 - 1 hour West of GMT</td>
<td>ZW12 - 12 hours West of GMT</td>
</tr>
<tr>
<td>ZW2 - 2 hours West of GMT</td>
<td>ZE12C - 12 3/4 hours East of GMT</td>
</tr>
<tr>
<td>ZW3 - 3 hours West of GMT</td>
<td>ZE12 - 12 hours East of GMT</td>
</tr>
<tr>
<td>NST - Newfoundland Standard Time</td>
<td>ZE11B - 11 1/2 hours East of GMT</td>
</tr>
<tr>
<td>AST - Atlantic Standard Time</td>
<td>ZE11 - 11 hours East of GMT</td>
</tr>
<tr>
<td>EST - Eastern Standard Time</td>
<td>ZE10B - 10 1/2 hours East of GMT</td>
</tr>
<tr>
<td>CST - Central Standard Time</td>
<td>ZE10 - 10 hours East of GMT</td>
</tr>
<tr>
<td>MST - Mountain Standard Time</td>
<td>ZE9B - 9 1/2 hours East of GMT</td>
</tr>
<tr>
<td>PST - Pacific Standard Time</td>
<td>ZE9 - 9 hours East of GMT</td>
</tr>
<tr>
<td>YST - Yukon Standard Time</td>
<td>ZE8 - 8 hours East of GMT</td>
</tr>
<tr>
<td>ZW9B - 9 1/2 hours West of GMT</td>
<td>ZE7 - 7 hours East of GMT</td>
</tr>
<tr>
<td>HST - Alaska-Hawaii Standard Time</td>
<td>ZE6 - 6 hours East of GMT</td>
</tr>
<tr>
<td>ZE6B - 6 1/2 hours East of GMT</td>
<td>ZE6C - 5 3/4 hours East of GMT</td>
</tr>
<tr>
<td>MST - Mountain Standard Time</td>
<td>ZE5B - 5 1/2 hours East of GMT</td>
</tr>
<tr>
<td>PST - Pacific Standard Time</td>
<td>ZE5 - 0 hours East of GMT</td>
</tr>
<tr>
<td>ZW12 - 12 hours West of GMT</td>
<td>ZW12 - 12 hours West of GMT</td>
</tr>
<tr>
<td>ZW12C - 12 3/4 hours East of GMT</td>
<td>ZW12C - 12 3/4 hours East of GMT</td>
</tr>
<tr>
<td>ZW12B - 12 1/2 hours East of GMT</td>
<td>ZW12B - 12 1/2 hours East of GMT</td>
</tr>
<tr>
<td>ZW - 2 hours West of GMT</td>
<td>ZW - 2 hours West of GMT</td>
</tr>
<tr>
<td>ZW2 - 2 hours West of GMT</td>
<td>ZW2 - 2 hours West of GMT</td>
</tr>
<tr>
<td>ZW3 - 3 hours West of GMT</td>
<td>ZW3 - 3 hours West of GMT</td>
</tr>
<tr>
<td>ZW4 - 4 hours West of GMT</td>
<td>ZW4 - 4 hours West of GMT</td>
</tr>
<tr>
<td>ZW5 - 5 hours West of GMT</td>
<td>ZW5 - 5 hours West of GMT</td>
</tr>
<tr>
<td>ZW6 - 6 hours West of GMT</td>
<td>ZW6 - 6 hours West of GMT</td>
</tr>
<tr>
<td>ZW7 - 7 hours West of GMT</td>
<td>ZW7 - 7 hours West of GMT</td>
</tr>
<tr>
<td>ZW8 - 8 hours West of GMT</td>
<td>ZW8 - 8 hours West of GMT</td>
</tr>
<tr>
<td>ZW9 - 9 hours West of GMT</td>
<td>ZW9 - 9 hours West of GMT</td>
</tr>
<tr>
<td>ZW10 - 10 hours West of GMT</td>
<td>ZW10 - 10 hours West of GMT</td>
</tr>
<tr>
<td>ZW11 - 11 hours West of GMT</td>
<td>ZW11 - 11 hours West of GMT</td>
</tr>
<tr>
<td>ZW12 - 12 hours West of GMT</td>
<td>ZW12 - 12 hours West of GMT</td>
</tr>
<tr>
<td>ZW12C - 12 3/4 hours East of GMT</td>
<td>ZW12C - 12 3/4 hours East of GMT</td>
</tr>
<tr>
<td>ZW12B - 12 1/2 hours East of GMT</td>
<td>ZW12B - 12 1/2 hours East of GMT</td>
</tr>
<tr>
<td>ZW12B - 12 1/2 hours East of GMT</td>
<td>ZW12B - 12 1/2 hours East of GMT</td>
</tr>
<tr>
<td>ZW - 2 hours West of GMT</td>
<td>ZW - 2 hours West of GMT</td>
</tr>
<tr>
<td>ZW2 - 2 hours West of GMT</td>
<td>ZW2 - 2 hours West of GMT</td>
</tr>
<tr>
<td>ZW3 - 3 hours West of GMT</td>
<td>ZW3 - 3 hours West of GMT</td>
</tr>
<tr>
<td>ZW4 - 4 hours West of GMT</td>
<td>ZW4 - 4 hours West of GMT</td>
</tr>
<tr>
<td>ZW5 - 5 hours West of GMT</td>
<td>ZW5 - 5 hours West of GMT</td>
</tr>
<tr>
<td>ZW6 - 6 hours West of GMT</td>
<td>ZW6 - 6 hours West of GMT</td>
</tr>
<tr>
<td>ZW7 - 7 hours West of GMT</td>
<td>ZW7 - 7 hours West of GMT</td>
</tr>
<tr>
<td>ZW8 - 8 hours West of GMT</td>
<td>ZW8 - 8 hours West of GMT</td>
</tr>
<tr>
<td>ZW9 - 9 hours West of GMT</td>
<td>ZW9 - 9 hours West of GMT</td>
</tr>
<tr>
<td>ZW10 - 10 hours West of GMT</td>
<td>ZW10 - 10 hours West of GMT</td>
</tr>
<tr>
<td>ZW11 - 11 hours West of GMT</td>
<td>ZW11 - 11 hours West of GMT</td>
</tr>
<tr>
<td>ZW12 - 12 hours West of GMT</td>
<td>ZW12 - 12 hours West of GMT</td>
</tr>
<tr>
<td>ZW12C - 12 3/4 hours East of GMT</td>
<td>ZW12C - 12 3/4 hours East of GMT</td>
</tr>
<tr>
<td>ZW12B - 12 1/2 hours East of GMT</td>
<td>ZW12B - 12 1/2 hours East of GMT</td>
</tr>
<tr>
<td>MST - Mountain Standard Time</td>
<td>MST - Mountain Standard Time</td>
</tr>
<tr>
<td>PST - Pacific Standard Time</td>
<td>PST - Pacific Standard Time</td>
</tr>
<tr>
<td>YST - Yukon Standard Time</td>
<td>YST - Yukon Standard Time</td>
</tr>
<tr>
<td>ZW - 2 hours West of GMT</td>
<td>ZW - 2 hours West of GMT</td>
</tr>
<tr>
<td>ZW2 - 2 hours West of GMT</td>
<td>ZW2 - 2 hours West of GMT</td>
</tr>
<tr>
<td>ZW3 - 3 hours West of GMT</td>
<td>ZW3 - 3 hours West of GMT</td>
</tr>
<tr>
<td>ZW4 - 4 hours West of GMT</td>
<td>ZW4 - 4 hours West of GMT</td>
</tr>
<tr>
<td>ZW5 - 5 hours West of GMT</td>
<td>ZW5 - 5 hours West of GMT</td>
</tr>
<tr>
<td>ZW6 - 6 hours West of GMT</td>
<td>ZW6 - 6 hours West of GMT</td>
</tr>
<tr>
<td>ZW7 - 7 hours West of GMT</td>
<td>ZW7 - 7 hours West of GMT</td>
</tr>
<tr>
<td>ZW8 - 8 hours West of GMT</td>
<td>ZW8 - 8 hours West of GMT</td>
</tr>
<tr>
<td>ZW9 - 9 hours West of GMT</td>
<td>ZW9 - 9 hours West of GMT</td>
</tr>
<tr>
<td>ZW10 - 10 hours West of GMT</td>
<td>ZW10 - 10 hours West of GMT</td>
</tr>
<tr>
<td>ZW11 - 11 hours West of GMT</td>
<td>ZW11 - 11 hours West of GMT</td>
</tr>
<tr>
<td>ZW12 - 12 hours West of GMT</td>
<td>ZW12 - 12 hours West of GMT</td>
</tr>
<tr>
<td>ZW12C - 12 3/4 hours East of GMT</td>
<td>ZW12C - 12 3/4 hours East of GMT</td>
</tr>
<tr>
<td>ZW12B - 12 1/2 hours East of GMT</td>
<td>ZW12B - 12 1/2 hours East of GMT</td>
</tr>
</tbody>
</table>

**Variable Data** - This is the start of the variable data field. Use the offset and length fields to actually find and retrieve data from this area.

**List Formats**

**List Type 1**

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Binary(4)</td>
<td>Number of Items in List</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Binary(4)</td>
<td>Length of each item</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>Character(*)</td>
<td>Items in the list</td>
</tr>
</tbody>
</table>
List Type 2

TCP List Format

### Field Descriptions

**Length of Network Name** - This is the length of the Network Name data.

**Length of Port Name** - This is the length of the port name data.

**Offset to Network Name** - This is the offset, from the start of this list element, to where the network name can be found.

**Offset to next item** - This is the offset, from the start of the first element in the list, to where the next element can be found.

**Offset to Port Name** - This is the offset, from the start of this list element, to where the port name can be found.

**Port Enabled** - This is a flag that indicates if this Domino Server Port is enabled. Possible values are:

- '0' - Server port not enabled.
- '1' - Server port is enabled.
**Port Encrypted** - This indicates if the data sent through the port is encrypted. Possible values are:

* ENCRYPT
* NOENCRYPT

**TCP/IP Address** - This is the TCP/IP address for this Domino Server Port.

**Error Messages**

CPF24B4: Severe error while addressing parameter list.

CPF3CF1: Error code parameter not valid

CPF3CF2: Error(s) occurred during running of &1 API.

CPF3C1E: Required parameter &1 omitted.

CPF3C21: Format name &1 is not valid.

CPF3C24: Length of the receiver variable is not valid.

LNT0907: The server name specified, &1, is not valid.

---

**Set Domino Environment (QnninSetDominoEnv)**

QnninSetDominoEnv is an API provided by the QNNINLDS SRVPGM. This API will set the current jobs working environment into a state to allow the NotesInitExtended API to be called for a specific Domino server. Information such as, Servers data directory, current user, Domino executable path, etc. will be set correctly after calling QnninSetDominoEnv. This removes the burden from the caller from having to know this specific information about each Domino server when initializing the Notes API environment.

**Required Parameter Group:**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Action to Perform</td>
<td>I</td>
</tr>
<tr>
<td>2</td>
<td>Server name</td>
<td>I</td>
</tr>
<tr>
<td>3</td>
<td>Server name length</td>
<td>I</td>
</tr>
<tr>
<td>4</td>
<td>User Profile Action</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>Path Action</td>
<td>I</td>
</tr>
<tr>
<td>6</td>
<td>Error Code</td>
<td>I/O</td>
</tr>
</tbody>
</table>

**Service Program Name:** QNNINLDS

**Similar Commands:** SETDOMENV
### Authorities and Locks

**Default public authority**

*USE

#### Required Parameter Group

**Action to Perform**

INPUT; BINARY(4)

If this value is 0, the API will set the current jobs environment to a state ready for the NotesInitExtended() API call. Job attributes such as PATH environment variable, current working directory, current user, and library list may be changed by this API. If this value is 1, the API will reset the environment back to its original state. The QnninSetDominoEnv API must have been called previously with a value of 0 for this parameter in order to call it a second time with a value of 1.

**Server name**

INPUT; CHAR(255)

The name of the Domino server to retrieve information.

**Server name length**

INPUT; BINARY(4)

The length of the server name.

**User Profile Action**

INPUT; BINARY(4)

If this value is 0, the API will not change the current user for the job. If this value is 1, the current user for the job will be changed to a user profile that is required for the Domino server. It is recommended that this value be set so that authority and newly created objects while running in the Domino environment be set to the correct USRPRF.

**Path Action**

INPUT; BINARY(4)

If this value is 0, the PATH environment variable will be replaced with a PATH needed to run in the Domino environment. If this value is 1, the required Domino environment path elements will be added at the beginning of the existing PATH value. If this value is 2, the required Domino environment path elements will be added at the end of the existing PATH value.

**Note**  If specifying Path Action 2, and there is an existing PATH value that contains a notes.ini file for a different server, that file will be found before the notes.ini file for the server specified on this API call.
Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error Messages
CPF24B4: Severe error while addressing parameter list.
CPF3CF1: Error code parameter not valid
CPF3CF2: Error(s) occurred during running of &1 API.
CPF3C1E: Required parameter &1 omitted.
CPF3C24: Length of the receiver variable is not valid.
LNT0907: The server name specified, &1, is not valid.
LNT8891: Error resetting the Domino environment.

Get Domino Environment (QnninGetDominoEnv)
QnninGetDominoEnv is an API provided by the QNNINLDS SRVPGM. This API will retrieve information about the current jobs Domino server environment. The server for the current environment is determined by searching for the notes.ini file in the PATH environment variable. If the notes.ini file can not be found in the current PATH, this API will fail.

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Data buffer for Domino environment</th>
<th>O</th>
<th>Char(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Data buffer length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>3</td>
<td>Format name</td>
<td>I</td>
<td>Char(8)</td>
</tr>
<tr>
<td>4</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Service Program Name: QNNINLDS

Similar Commands: None

Authorities and Locks
Default public authority
*USE
**Required Parameter Group**

**Data buffer for Domino environment**

**OUTPUT; CHAR(*)**

This returns the information about a Domino environment. For the format, see "Data Buffer".

**Data buffer length**

**INPUT; BINARY(4)**

The length of the data buffer. The length must be at least big enough to hold the Bytes returned field and Bytes available field. Failure to provide enough room for the data will result in errors or incomplete data being returned.

**Format name**

**INPUT; CHAR(8)**

The name of the format used to retrieve the Domino environment information. You can use the following format names:

DENV0100

**Error code**

**I/O; CHAR(*)**

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

**Data Buffer**

For detailed descriptions of the fields in these tables, see Field Descriptions. The QnninDominoEnv definition has been supplied in the qnninlds.h file to assist mapping the data buffer to a C/C++ data structure.
DENV0100 Attribute Buffer

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Binary(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Binary(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Binary(4)</td>
<td>Length of server name</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Char(256)</td>
<td>Server Name</td>
</tr>
<tr>
<td>268</td>
<td>44</td>
<td>Binary(4)</td>
<td>Length of data directory path</td>
</tr>
<tr>
<td>272</td>
<td>110</td>
<td>Char(256)</td>
<td>Data directory name</td>
</tr>
<tr>
<td>528</td>
<td>210</td>
<td>Binary(4)</td>
<td>Length of run path</td>
</tr>
<tr>
<td>532</td>
<td>214</td>
<td>Char(256)</td>
<td>Run path</td>
</tr>
<tr>
<td>788</td>
<td>314</td>
<td>Char(10)</td>
<td>Server's runtime library</td>
</tr>
<tr>
<td>798</td>
<td>31</td>
<td>Char(10)</td>
<td>Server's user profile</td>
</tr>
<tr>
<td>808</td>
<td>328</td>
<td>Char(16)</td>
<td>Server's release</td>
</tr>
<tr>
<td>824</td>
<td>338</td>
<td>Char(16)</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Field Descriptions

Bytes available: The length of data that could be returned by this API. This value is greater than "Bytes returned" when the length of the receiving variable is too small, causing truncation of data.

Bytes returned: The length of data returned in this structure. This value includes this and all following fields. If the data is truncated because the receiver variable is not large enough to hold all of the data available, this value will be less than the bytes available value.

Length of server name: The length, in bytes, of the servers name returned in the server name field.

Server Name: The Domino server's name determined from the current jobs environment.

Length of data directory path: The length, in bytes, of the data directory returned in the Data directory name field.

Data directory name: The Domino server's data directory determined from the current jobs environment.

Length of run path: The length, in bytes, of the Run path returned in the Run path field.

Run path: A PATH environment component that contains the Domino server's executable information.

Server's runtime library: The library that contains the Domino server's executable binary objects.
Server's user profile: The user profile that the Domino server jobs will run under.

Server's release: A character representation in CCSID 37, of the Domino server's release.

Reserved: Reserved field.

Error Messages
CPF24B4: Severe error while addressing parameter list.
CPF3CF1: Error code parameter not valid
CPF3CF2: Error(s) occurred during running of &1 API.
CPF3C1E: Required parameter &1 omitted.
CPF3C21: Format name &1 is not valid.
CPF3C24: Length of the receiver variable is not valid.
LNT0907: The server name specified, &1, is not valid.
LNT8895: Domino server environment information cannot be retrieved.

List Domino Release Information (QnninListDominoRlsI)
QnninListDominoRlsI is an API provided by the QNNINLDS SRVPGM. This API will retrieve a list of installed Domino releases on the current iSeries system.

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Data buffer for Domino releases</th>
<th>O</th>
<th>Char(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Data buffer length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>3</td>
<td>Format name</td>
<td>I</td>
<td>Char(8)</td>
</tr>
<tr>
<td>4</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Service Program Name: QNNINLDS

Similar Commands: None

Authorities and Locks
Default public authority

*USE
Required Parameter Group

Data buffer for Domino releases

OUTPUT; CHAR(*)

This returns the list of installed Domino releases. For the format, see "Data Buffer".

Data buffer length

INPUT; BINARY(4)

The length of the data buffer. The length must be at least big enough to hold the Bytes returned field and Bytes available field. Failure to provide enough room for the data will result in errors or incomplete data being returned.

Format name

INPUT; CHAR(8)

The name of the format used to retrieve the Domino environment information. You can use the following format names:

DRLS0100

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Data Buffer

For detailed descriptions of the fields in these tables, see Field Descriptions. The QnninListDominoRls definition has been supplied in the qnninlds.h file to assist mapping the data buffer to a C/C++ data structure.

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Binary(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Binary(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Binary(4)</td>
<td>Offset to release entries</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Binary(4)</td>
<td>Number of release entries</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Binary(4)</td>
<td>Length of each release entry</td>
</tr>
</tbody>
</table>
Field Descriptions

**Bytes available:** The length of data that could be returned by this API. This value is greater than "Bytes returned" when the length of the receiving variable is too small, causing truncation of data.

**Bytes returned:** The length of data returned in this structure. This value includes this and all following fields. If the data is truncated because the receiver variable is not large enough to hold all of the data available, this value will be less than the bytes available value.

**Offset to release entries:** The offset, in bytes, from the beginning of the buffer to the array of Domino release information (See below).

**Number of release entries:** The number of release entries returned in the array of Domino release information.

**Length of each release entry:** The length, in bytes, of each Domino release array element.

**DRLS0100 Release Information array element:** The DominoRls100 definition has been supplied in the qnninlds.h file to assist mapping the release information to a C/C++ data structure.

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Binary(4)</td>
<td>Length of run path</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Char(256)</td>
<td>Run path</td>
</tr>
<tr>
<td>260</td>
<td>104</td>
<td>Char(10)</td>
<td>Run time library</td>
</tr>
<tr>
<td>270</td>
<td>10</td>
<td>Char(16)</td>
<td>Release</td>
</tr>
<tr>
<td>286</td>
<td>11</td>
<td>Char(7)</td>
<td>Product ID</td>
</tr>
<tr>
<td>293</td>
<td>125</td>
<td>Char(5)</td>
<td>Product Option</td>
</tr>
<tr>
<td>298</td>
<td>12A</td>
<td>Char(16)</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Field Descriptions

**Length of run path:** The length, in bytes, of the run path returned in the Run path field.

**Run path:** A PATH environment component that contains a Domino release's executable information.

**Run time library:** The library that contains the Domino release's executable binary objects.

**Release:** A character representation in CCSID 37, of the Domino release.

**Product ID:** The licensed program product ID of the Domino release.

**Product Option:** The licensed program option of the Domino release.

**Reserved:** Reserved field.
Error Messages
CPF24B4: Severe error while addressing parameter list.
CPF3CF1: Error code parameter not valid
CPF3CF2: Error(s) occurred during running of &1 API.
CPF3C1E: Required parameter &1 omitted.
CPF3C21: Format name &1 is not valid.
CPF3C24: Length of the receiver variable is not valid.
LNT0907: The server name specified, &1, is not valid.

Get Notes.ini Value(QnninGetIniValue)
QnninGetIniValue and QnninGetIniValuez are APIs provided by the QNNINLDS SRVPGM. These APIs will retrieve a value from a Domino server's notes.ini file. These APIs will behave similarly to the Domino OSGetEnvironmentString API.

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Server name</th>
<th>I</th>
<th>Char(255)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Server name length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>3</td>
<td>Notes.ini variable name</td>
<td>I</td>
<td>Char(*)</td>
</tr>
<tr>
<td>4</td>
<td>Notes.ini variable name length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>5</td>
<td>Return Buffer</td>
<td>O</td>
<td>Char(*)</td>
</tr>
<tr>
<td>6</td>
<td>Return Buffer Length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>7</td>
<td>Return Bytes Available</td>
<td>O</td>
<td>Binary(*)</td>
</tr>
<tr>
<td>8</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Service Program Name: QNNINLDS

Similar Commands: None

Authorities and Locks
Default public authority

*USE
Required Parameter Group

Server name
INPUT; CHAR(255)
The name of the Domino server to retrieve information.

Server name length
INPUT; BINARY(4)
The length of the server name.

Notes.ini variable name
INPUT; CHAR(*)
The variable name for the entry in the notes.ini to be retrieved. Notes.ini entries are in the form of variable=value; The variable name is expected to be in CCSID 37.

Notes.ini variable name length
INPUT; BINARY(4)
The length of the variable name supplied in the notes.ini variable name parameter.

Return Buffer
OUTPUT; CHAR(*)
The value associated with the notes.ini variable name will be returned in this buffer. The value will be returned in a CCSID 37 character array.

Return Buffer Length
INPUT; BINARY(4)
The size of the buffer that will be used to return the notes.ini value.

Return Bytes Available
OUTPUT; BINARY(*)
The number of bytes returned in the Return Buffer. If this value is larger than Return Buffer Length, then only Return Buffer Length bytes will be returned.

Error code
I/O; CHAR(*)
The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.
**Error Messages**

CPF24B4: Severe error while addressing parameter list.
CPF3CF1: Error code parameter not valid
CPF3CF2: Error(s) occurred during running of &1 API.
CPF3C1E: Required parameter &1 omitted.
CPF3C24: Length of the receiver variable is not valid.
LNT0907: The server name specified, &1, is not valid.
LNT8896: Error occurred when trying to set or get Domino configuration value.

---

**Get Notes.Ini Value - zero terminated (QnninGetIniValuez)**

This API is identical to the QnninGetIniValue API except that the input string parameters are entered as null terminated strings. This eliminates the parameters associated with specifying lengths of character strings. This API is useful when calling from an environment that may restrict the number of parameters that can be passed.

**Required Parameter Group:**

<table>
<thead>
<tr>
<th></th>
<th>Parameter</th>
<th>Mode</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Server name</td>
<td>I</td>
<td>Char(255)</td>
</tr>
<tr>
<td>2</td>
<td>Notes.ini variable name</td>
<td>I</td>
<td>Char(*)</td>
</tr>
<tr>
<td>3</td>
<td>Return Buffer</td>
<td>O</td>
<td>Char(*)</td>
</tr>
<tr>
<td>4</td>
<td>Return Buffer Length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>5</td>
<td>Return Bytes Available</td>
<td>O</td>
<td>Binary(*)</td>
</tr>
<tr>
<td>6</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

See QnninGetIniValue for parameter descriptions. Input Char(*) and Char(n) parameters are expected to be null terminated.

---

**Get Server Document Item - zero terminated (QnninGetServerDocItemz)**

This API is identical to the QnninGetServerDocItem API except that the input string parameters are entered as null terminated strings. This eliminates the parameters associated with specifying lengths of character strings. This API is useful when calling from an environment that may restrict the number of parameters that can be passed.

---

74 Domino 6 for iSeries Application Development Guide
### Required Parameter Group

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Server name</td>
<td>I</td>
<td>Char(255)</td>
</tr>
<tr>
<td>2</td>
<td>Server document item name</td>
<td>I</td>
<td>Char(*)</td>
</tr>
<tr>
<td>3</td>
<td>Return Buffer</td>
<td>O</td>
<td>Char(*)</td>
</tr>
<tr>
<td>4</td>
<td>Return Buffer Length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>5</td>
<td>Return Bytes Available</td>
<td>O</td>
<td>Binary(*)</td>
</tr>
<tr>
<td>6</td>
<td>Return Buffer data type</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>7</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

For parameter descriptions, see QnninGetServerDocItem. Input Char(*) and Char(n) parameters are expected to be null terminated.

---

### Set Server Document Item (QnninSetServerDocItem)

QnninSetServerDocItem and QnninSetServerDocItemz are APIs provided by the QNNINLDS SRVPGM. These APIs will change an item value in a Domino server's server document found in the Domino directory (names.nsf file). These APIs will behave similarly to the Domino NSFItemSetText or NSFItemSetNumber APIs. These APIs can also be used to remove an item from the server document.

#### Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Server name</td>
<td>I</td>
<td>Char(255)</td>
</tr>
<tr>
<td>2</td>
<td>Server name length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>3</td>
<td>Server document item name</td>
<td>I</td>
<td>Char(*)</td>
</tr>
<tr>
<td>4</td>
<td>Server document item name length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>5</td>
<td>New Value</td>
<td>I</td>
<td>Char(*)</td>
</tr>
<tr>
<td>6</td>
<td>New Value Length</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>7</td>
<td>List Processing Flags</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>8</td>
<td>New Value data type</td>
<td>I</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>9</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

**Service Program Name:** QNNINLDS

**Similar Commands:** None

**Authorities and Locks**

**Default public authority**

*USE*
Required Parameter Group

Server name
INPUT; CHAR(255)
The name of the Domino server to set information.

Server name length
INPUT; BINARY(4)
The length of the server name.

Server document item name
INPUT; CHAR(*)
The item name for the entry in the server document to be set. The variable name is expected to be in CCSID 37.

Server document item name length
INPUT; BINARY(4)
The length of the item name supplied in the Server document item name parameter.

New Value
INPUT; CHAR(*)
The value associated with the server document item will be set to this value. This parameter may also contain a value to be removed from an existing list entry in the notes.ini if the List Processing Flags (see below) is set to (2) - Remove.

New Value Length
INPUT; BINARY(4)
The size of the New Value. If this parameter is 0, then the item in the server document is cleared.

List Processing Flags
INPUT; BINARY(4)
This flag indicates how to operate on items that are in the form of a list. The valid values are:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Replace</td>
<td>Replace the value that may currently be set.</td>
</tr>
<tr>
<td>1</td>
<td>Append</td>
<td>Append the new value to an existing list.</td>
</tr>
<tr>
<td>2</td>
<td>Remove</td>
<td>Remove the value from a list. The value in New Value will be removed from the list associated with the item name.</td>
</tr>
</tbody>
</table>

76 Domino 6 for iSeries Application Development Guide
New Value data type

INPUT; BINARY(4)

This flag indicates what data type format the new value is in. The valid values are:

<table>
<thead>
<tr>
<th></th>
<th>Text Data</th>
<th>Float Data</th>
<th>Integer Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New Value is a CCSID 37 CHAR(*)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>New Value is a C/C++ compatible &quot;double&quot; data type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>New Value is a BINARY(4) value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error Messages

CPF24B4: Severe error while addressing parameter list.
CPF3CF1: Error code parameter not valid
CPF3CF2: Error(s) occurred during running of &1 API.
CPF3C1E: Required parameter &1 omitted.
CPF3C24: Length of the receiver variable is not valid.
LNT0907: The server name specified, &1, is not valid.
LNT8896: Error occurred when trying to set or get Domino configuration value.

Set Server Document Item - zero terminated (QnninSetServerDocItemz)

This API is identical to the QnninSetServerDocItem API except that the input string parameters are entered as null terminated strings.

This eliminates the parameters associated with specifying lengths of character strings. This API is useful when calling from an environment that may restrict the number of parameters that can be passed.
**Required Parameter Group**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Char(255)</td>
<td>I</td>
</tr>
<tr>
<td>Server document item name</td>
<td>Char(*)</td>
<td>I</td>
</tr>
<tr>
<td>New Value</td>
<td>Char(*)</td>
<td>I</td>
</tr>
<tr>
<td>List Processing Flags</td>
<td>Binary(4)</td>
<td>I</td>
</tr>
<tr>
<td>New Value data type</td>
<td>Binary(4)</td>
<td>I</td>
</tr>
<tr>
<td>Error Code</td>
<td>Char(*)</td>
<td>I/O</td>
</tr>
</tbody>
</table>

For parameter descriptions, see QnninSetServerDocItem for parameter descriptions. Input Char(*) and Char(n) parameters are expected to be null terminated.

**Note** To remove a server document item, the New Value string should be set to a null string. This behavior is different in that there is no New Value length parameter that can be set to 0 to indicate an item should be cleared. Instead, the New Value length is determined by the fact that the first character in the string is null.

---

**Using the Management Collection Object API to read Domino collection data**

Administrators running Domino 6 for iSeries on an iSeries server running Version 5 Release 2 of OS/400 can use the Collection Services of OS/400 to collect performance data about the Domino servers running on their iSeries servers. When Collection Services is configured to collect Domino data, each interval contains contain statistics for all active Domino servers that are running the COLSRV400 addin task.

The full set of Domino statistics, (that is, the statistics seen in the Domino console command "show stat") are stored in the Management Collection Object (MCO). This data can be exported from the MCO to a DB2 UDB for iSeries database using the OS/400 Create Performance Data command. As an alternative, you could use the APIs described in the following sections to export the MCO data into a tool of your choice. The following sections provide examples and information on accessing the Domino data from an MCO.
For more information about iSeries Collection Services and the Management Collection Object in the V5R2 iSeries Information Center, which is available at the following URL:

http://www.ibm.com/eserver/iseries/infocenter

Click **Systems Management --> Performance --> Applications for Performance Management --> Collection Services:**

For more information about configuring Domino to collect data for Collection Services, see *Installing and Managing Domino 6 for iSeries* (i400help.pdf or i400help.nsf).

**How Domino statistics are stored**

Domino statistics are made up of words, or name fragments, separated by periods, such as Server. Users.Peak. In this case, the words Server, Users and Peak are name fragments that comprise Peak Users statistic for a particular server.

Each name fragment is stored in an entry, or Node that is alphabetically sorted. The Server data section is made up of entries of these nodes. The position of each fragment is referred to as a level, with level 1 indicated by the name fragment furthest to the left. For example, the levels of the Server. Users. Peak statistic are:

<table>
<thead>
<tr>
<th>Name Fragment</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>1</td>
</tr>
<tr>
<td>Users</td>
<td>2</td>
</tr>
<tr>
<td>Peak</td>
<td>3</td>
</tr>
</tbody>
</table>

The statistics collected during an interval varies based on the number of servers that were active at that particular time. The statistics are stored in a tree format with the intention of reducing file size and reduced search times.

**Note** All character strings in the data are stored in a superset of ASCII known as LMBCS. You can work with the data as ASCII (code page 850). The example programs that follow this section illustrate converting this data into EBCDIC.

The statistics interval data can contain statistics for up to 99 Domino servers. The beginning of the interval data will list the number of servers and the total length of the file:

<table>
<thead>
<tr>
<th>File Offset</th>
<th>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BIN2</td>
<td>Number of Servers (stored in this interval)</td>
</tr>
<tr>
<td>2</td>
<td>BIN8</td>
<td>Total file length</td>
</tr>
<tr>
<td>10</td>
<td>CHAR 6</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Each server contains a header section of information. All headers follow one after another for each server. Here is the first server's header:
If there is another server in this interval, its header follows immediately after the previous server (In this case, offset 113). The pattern continues for each additional server. Use the "Number of Servers" field to determine when to stop reading.

To access a server's data, follow the "Absolute File Offset to start of this server's data" field in the header. The data will start at this offset for the start of the file. You do not need to know where the data for the server ends because of the way you access it. The following example illustrates accessing the data.

### Locating a statistic

The following example illustrates the process you could use when looking for a statistic (for example, peak users) for a particular server.

1. Start with the first node, Domino, in the first node of the interval data.
2. Locate "Server." The "Domino" node will have a pointer to the "Next Same Level" node that is the next name fragment on the same level.
3. Next, consider the next node of "Platform."
4. Try the next node and find "Server."
   
   **Note** If you found "Time" instead of Server, then there are not any Server.xxxx.xxx statistics in this particular interval data.
5. Once "Server" is found, look for a pointer for the "Next Lower Level" of that statistic's name (adding 1 to the current level). This would be a statistic name at Level 2 (that is, Server.<here>).
6. Now look to see if there is a "Next Level" pointer for "Server." Since a statistic only named "Server" does not exist, there should be a "Next Level" pointer. The offset may point to a node named "Restart." Since you are looking for "Users" for the next level, look to the next "Same Level" pointer of "Restart." If the next node is named "Users," you have located Users portion of this statistic. You now have found Server.Users.
7. The last fragment to find is "Peak." At the "Users" node, continue to the "Next Lower Level" pointer to go to level 3. This could lead you to a node called "Peak." If this is the case, then you have found the statistic.

Since you have a full statistic name, the value of the statistic should be stored in this node. If there is a value in this node, the field "Data Value Offset" will be any numeric value other than 0. The type of data represented by the statistic is stored (It will be a double 8 Byte floating point number, a long integer, a text string, or a date time represented in a text string). Based on this example, the format of each node follows:

<table>
<thead>
<tr>
<th>Offset</th>
<th>DataType Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BIN4</td>
<td>Offset of this Node (Relative from start of this server's node data section, as are all offsets in this node)</td>
</tr>
<tr>
<td>4</td>
<td>BIN4</td>
<td>Offset of 'Next Same Level' Node</td>
</tr>
<tr>
<td>8</td>
<td>BIN4</td>
<td>Offset of 'Next Lower Level' Node</td>
</tr>
<tr>
<td>12</td>
<td>BIN4</td>
<td>Offset to name fragment of this node</td>
</tr>
<tr>
<td>16</td>
<td>BIN4</td>
<td>Offset to data value of this node (0 if there is none)</td>
</tr>
<tr>
<td>20</td>
<td>BIN2</td>
<td>Name fragment length in characters</td>
</tr>
<tr>
<td>22</td>
<td>BIN2</td>
<td>Data Value Size (If there is one in this node. Size in bytes of datatype, ex: BIN4 is 4 Bytes)</td>
</tr>
<tr>
<td>24</td>
<td>BIN2</td>
<td>Type of Data represented by statistic</td>
</tr>
<tr>
<td>26</td>
<td>CHARxx</td>
<td>statistic fragment name</td>
</tr>
<tr>
<td>xx (Starts right after fragment name)</td>
<td>datatype xx, size yy</td>
<td>statistic value</td>
</tr>
</tbody>
</table>

**Offset of this Node**: Offset relative to the start of nodes data for this server.

**Offset of 'Next Same Level' node**: Offset to the next node that contains a new statistic name fragment at the same level of this statistic name fragment. Value is 0 if there is none. (For example, Server.Users going to Server.Time)

**Offset of 'Next Lower Level' node**: Offset to the next node that contains a new additional statistic name fragment at the end of the current statistic name fragment. Value is 0 if there is none.

**Offset of 'Next Lower Level' node**: Offset to the next node that contains a new additional name fragment which can be appended to the end of the current statistic name. (For example, if we are on the node 'Users', and the current statistic is Server.Users, the next lower level fragment could be "Peak". That would create "Server.Users.Peak" at the next node.) If there are no additional fragments that can be appended, the value of the offset is 0.

Chapter 4: Domino for iSeries APIs  81
**Offset to name fragment** - Offset to start of name fragment represented in this node (For example, this could be 'Users' in Server.Users).

**Offset to data value** - Offset to data value of the statistic, if you are at the final fragment. A value here signals the end of a statistic name. (For example, if a fragment is 'Users' and the statistic is "Server.Users = 40", you will store a '40' to store as the data). The value is 0 if there is no data with this fragment.

**Name Fragment Size** - Size of name fragment in characters.

**Data Value Size** - Size of datatype (or characters if text) representing the value of the statistic, if there is one with this node. A value of 0 means no value.

**Type of Data:**
- 0 LONG (Signed BIN4 Integer)
- 1 TEXT (CHAR string)
- 2 TIMEDATE (Format is a char string of YYYYMMDDHHMMSS)
- 3 DOUBLE (8-Byte Floating point number, same structure as a 'double' in C)

**Statistic Fragment Name Data** - Characters representing the name fragment.

**Statistic Value Data** - Statistic value data in it's appropriate form.

**Example set of Nodes:**
The following statistics are contained in the node table below:

- Domino.db.cache.size = 10
- Domino.db.hits = 20
- Domino.db.hits.persecond = 30
- Domino.db.size = 40
- Server.name = "max"
- Server.time = "20010911123045"
- Server.users.active = 50
- Server.users.total = 60
Example: Locating the Server.Users.Active

To locate a statistic, for example, "Server.Users.Active:

1. Start at Node 1. The node fragment name is "Domino." To find "Server," skip to the "Next Same Level." Follow file offset to 237 (Node 8).
2. Look at Node 8. The level name is "Server." You should now have level 1 of the name. The next level is "Users." Look at the next lower level at offset 269 (Node 9).
3. Look at Node 9. Level name is "Name" and you are looking for "Users," so skip to the "Next Same Level," Offset 302, Node 10.

<table>
<thead>
<tr>
<th>Node</th>
<th>Offset to Next Same Level</th>
<th>Offset to Next Lower Level</th>
<th>Name Fragment Size</th>
<th>Data Type</th>
<th>Name Data</th>
<th>Data (End Of node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>237</td>
<td>29</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>0</td>
<td>57</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>57</td>
<td>122</td>
<td>88</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>88</td>
<td>0</td>
<td>114</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>122</td>
<td>199</td>
<td>156</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>156</td>
<td>0</td>
<td>182</td>
<td>9</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>199</td>
<td>0</td>
<td>225</td>
<td>4</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>237</td>
<td>0</td>
<td>269</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>269</td>
<td>302</td>
<td>295</td>
<td>4</td>
<td>3</td>
<td>1 (text)</td>
</tr>
<tr>
<td>10</td>
<td>302</td>
<td>346</td>
<td>328</td>
<td>4</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>346</td>
<td>0</td>
<td>377</td>
<td>5</td>
<td>0</td>
<td>users</td>
</tr>
<tr>
<td>12</td>
<td>377</td>
<td>413</td>
<td>403</td>
<td>6</td>
<td>4</td>
<td>0 (long)</td>
</tr>
<tr>
<td>13</td>
<td>413</td>
<td>0</td>
<td>439</td>
<td>5</td>
<td>4</td>
<td>0 (long)</td>
</tr>
</tbody>
</table>

Chapter 4: Domino for iSeries APIs  83
5. Look at Node 11. Level name is "Users." We now have levels 1 and 2. The next level to find is "Active." Now look at the next lower level at offset 377 (Node 12)

6. Look at Node 12. Level name is "active." You now have all three levels of the full name.

7. The "Offset to Data" of Node 12 field is not 0, so there is a value associated with "Server.Users.Active." The "Data Type" is 0, or a long integer, and it's "Data Size" is 4 bytes. To get the value, read 4 bytes of memory at "Offset to Data" 444. This value represents 50 in long integer format.

Note A node can have a data value but is not the last level of the statistic name. Consider 2 statistics named "Domino.db.hits" and "Domino.db.hits.persecond". Level 3, "hits", will have a data value since it completes the statistic name "Domino.db.hits", and will contain the value for that statistic. However, its "Next Lower Level" also has a pointer to the "persecond" node. Continuing to that node will then have the value of "Domino.db.hits.persecond". When the "Next Lower Level" of a node is 0, that means there are no more name fragments to add to the right of the statistic name (in this case, Domino.db.hits.persecond).

Example Code:
In the following examples, you can choose to use the most recent MCO or you can explicitly specify the name of a MCO. To find the names of MCOs available on the system, use iSeries Navigator, and click Configuration and Service -> Collection Services. You may see a number of entries in the right pane, with names like 'Q150151308'. Each of these entries is a MCO. The library of the object can be found by right clicking on the entry and selecting 'Properties'. Look for the section "Location: /Qsys.lib/Qmpgdata.lib". This example is in the QMPGDATA lib. The object type of these files are *MGTCOL.

The following code examples can be used to open a *MGTCOL object, and retrieve any Domino statistic, or print all statistics for all servers. The examples will use the first interval in the MCO. The MCO itself will typically have 24 hours worth of intervals. See the MCO API documentation in the iSeries Information Center on how to access other intervals within the MCO.

Note These files, including the header, are only provided as an example illustrating how the data is laid out. These files must be compiled and run on an iSeries server running OS/400 V5R2 or later.

FindStat: Finds and prints the value of 1 statistic on 1 server for 1 interval.
Parameters:

1. (char string) Domino Server Name (case sensitive)
2. (char string) Statistic name
3. (char string) Fully qualified *MGTCOL object and library, or "ACTIVE" to retrieve the current MCO. Object is first 10 characters, and library is the second 10 characters.

To use current MCO:

```
call qgpl/findstat parm('MAILSRV1', 'Server.Users.Peak', 'ACTIVE')
```

To use any MCO by name:

```
call qgpl/findstat parm('MAILSRV1', 'Server.Users.Peak', 'Q150151308QMPGDTA')
```

PrintAll: Prints all statistics from all servers for 1 interval.

Parameters:

(char string) Fully qualified *MGTCOL object and library, or "ACTIVE" to retrieve the current MCO. Object is first 10 characters, and library is the second 10 characters.

To use current MCO:

```
call qgpl/printall parm('ACTIVE')
```

To use any MCO by name:

```
call qgpl/printall parm('Q150151308QMPGDTA ')
```

**Note**  If you choose 'ACTIVE', and Collection Services is not running or is not collecting Domino data, you may receive an error.
Index

A
Agent, 15, 20
calling program, 15, 28
Java, 20, 22
API, 5, 45, 46
32-bit handles, 11
ASCII and EBCDIC, 12
C considerations, 14
definitions or cancel handlers, 11
global operators, 11
location of header files, 14
PATH environment variable, 10
server add-ins, 11
thread safety, 9
APPDEV.NSF, 17
Application development
API handles, 11
ASCII and EBCDIC, 12
calling programs, 28
definitions or cancel handlers, 11
general considerations, 5
global operators, 11
location of header files, 14
LotusScript considerations, 24
PATH environment variable, 10
server add-ins, 11
thread safety, 9
ASCII, 12, 13, 16

C
C API, 5, 15
accessing header files, 17
ASCII compilation, 16
ASCII-to-EBCDIC conversion, 12
compiler requirements, 14
compiling on AS/400, 17
general considerations, 5
C++ API, 5
EBCDIC requirement, 12
general considerations, 5
Cancel handler, 11
CHKPATH, 23
CLASSPATH environment variable, 20

Command
ADENVVVAR, 10
ADDLIBLE, 17, 20
ADDLNK, 11, 17
CPTYOSTMF, 17
Sbmjob, 20
WRKRDBDIRE, 32
Compiler
C API, 14
Connection add-ins, 11
thread safety, 9

D
Database integration
@Db functions, 37
errors messages, 43
LS:DO, 37, 38
methods, 31
relational database entry, 32
remote connections, 41
DB2 UDB for iSeries, 32
ways of accessing, 31
DB2/UDB for iSeries, 33
DECS, 31
Domino collection data, 78
Drive
shared, 17

E
EBCDIC, 12, 13
Environment variable
adding, 10
CLASSPATH, 20
PATH, 10, 20
Examples
setting environment variable, 10
Exception handler, 11

F
FTP, 17

G
Get Domino Environment, 66
Get Notes.ini Value, 72
Get Notes.Ini Value - zero terminated, 74
Get Server Document Item - zero terminated, 74

H
Header files
  accessing from compiler, 17
  where to find, 14
Hook driver, 30

J
Java, 5, 20, 22, 23
  AS/400 software required, 20
  calling program, 28
  general considerations, 5
  restrictions for running agents, 22
  setting up to run, 20
JavaMaxHeapSize, 22

L
LIBASCII, 13
Library
  QNOTESAPI, 14
  QNOTESCRIPT, 14
  QNOTESLSKT, 14
List Domino Release Information, 69
List Domino Servers, 46
LMBCS, 12, 16
LotusScript
  considerations, 24
LS:DO
  calling program, 28
  differences, 38
  error messages, 43
  performance, 37
  prestarted jobs, 38
  remote connections, 41
  requirements, 37

M
Management Collection Object, 78
MCO, 78
Message
  database integration, 43

N
NotesPump, 32

P
PATH environment variable, 10, 20
Prestarted job, 38

Q
QnninGetDominoEnv, 45, 64, 66
QnninGetIniValue, 45, 72
QnninGetIniValuez, 45, 74
QnninGetServerDocItem, 45
QnninGetServerDocItemz, 45
QnninListDominoRlsI, 45, 69
QnninListDominoServers, 45, 46
QnninRtvDominoServerAttr, 45, 53
QnninRtvDominoServerI, 45, 49
QnninSetDominoEnv, 45
QnninSetIniValue, 45
QnninSetIniValuez, 45
QnninSetServerDocItem, 45, 75
QnninSetServerDocItemz, 45, 74, 77
QNOTES user profile
  running application, 6
QSYS.LIB file system
  location of header files, 14

R
Relational database name, 32
Remote connection, 41
Restrictions
  exception or cancel handlers, 11
  global operators, 11
  Java agents, 22
  LS:DO, 38
  thread safety, 9
Retrieve Domino Server Attributes, 53
Retrieve Domino Server Information, 49

S
Server Information, 45
Set Domino Environment, 64
Set Server Document Item, 75
Set Server Document Item - zero terminated, 77
Setup, 1
  Java, 20
SQL, 38
Symbolic link, 11, 30
T
Teraspace, 12
Thread safety, 9

U
User profile
QNOTES, 6

Z
Zip file, 17