

Lotus. Domino®



Version 7
for i5/OS™



Application Development Guide

Disclaimer; No Warranty

THIS INFORMATION AND ALL OTHER DOCUMENTATION (IN PRINTED OR ELECTRONIC FORM) ARE PROVIDED FOR REFERENCE PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THIS INFORMATION, THIS INFORMATION AND ALL OTHER DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT ANY WARRANTY WHATSOEVER AND TO THE MAXIMUM EXTENT PERMITTED, LOTUS AND IBM DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SAME. LOTUS AND IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, DIRECT, INDIRECT, CONSEQUENTIAL OR INCIDENTAL DAMAGES, ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS INFORMATION OR ANY OTHER DOCUMENTATION. NOTWITHSTANDING ANYTHING TO THE CONTRARY, NOTHING CONTAINED IN THIS INFORMATION OR ANY OTHER DOCUMENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM LOTUS AND IBM (OR THEIR SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF THIS SOFTWARE.

Under the copyright laws, neither this documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of IBM Corporation, except in the manner described in the documentation or the applicable licensing agreement governing the use of the software.

© Copyright IBM Corporation 1998, 2005

All rights reserved. **Printed in the United States.**

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GS ADP Schedule Contract with IBM Corp.

Lotus Software, IBM Software Group
One Rogers Street
Cambridge, MA 02142

Trademarks

iSeries, iSeries Client Access Express, eServer, and i5/OS are trademarks and IBM, Operating System/400, and PowerPC are registered trademarks of International Business Machines Corporation. Domino, Sametime Connect, and QuickPlace are trademarks and Lotus, Lotus Notes, and Sametime are registered trademarks of Lotus Development Corporation and/or IBM Corporation in the United States, other countries, or both. Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. MeetingPlace is a trademark of Latitude Communications, Inc. Microsoft, NetMeeting, Outlook, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. Pentium is a registered trademark of Intel Corporation in the United States, other countries, or both. Other product and company names mentioned herein may be the trademarks or registered trademarks of their respective owners.

Contents

1 Application Development on Domino for i5/OS	1
About this guide	1
System Requirements	2
Additional resources for Domino application development	2
2 Application development considerations	5
General considerations for using C, C++, and Java API	5
Obtaining the C, C++ and Java API	6
Running under the QNOTES user profile	6
Threading considerations for i5/OS	9
Setting the PATH environment variable	10
Considerations for multi-versioning	11
Starting server add-ins	12
Restriction on APIs in exception or cancel handlers	12
Restriction on using global operators	13
Using API handles	13
Enabling your application for teraspace	13
ASCII-to-EBCDIC conversion	14
Where to find the header files on i5/OS	15
Considerations for C APIs	15
Calling C programs from Domino applications or agents	16
Considerations for developing applications that use Java	21
Using Java in Domino Agents on i5/OS	22
JavaMaxHeapSize *NOMAX	24
JavaOS400CHKPATH	24
Considerations for LotusScript functions on i5/OS ..	26
Using the LotusScript Declare Function statement to call C functions	29
About adding hook drivers	31
3 Integrating Notes and DB2/UDB for iSeries Data	33
Setup required to access DB2 UDB for iSeries ..	34
About authority when Domino applications access DB2 UDB for iSeries	35
4 Domino for i5/OS APIs	45
Domino for i5/OS server APIs	45
List Domino Servers (QnninListDominoServers) ..	46
Retrieve Domino Server Information (QnninRtvDominoServerI)	49
Retrieve Domino Server Attributes (QnninRtvDominoServerAttr)	52
Set Domino Environment (QnninSetDominoEnv) ..	62
Get Domino Environment (QnninGetDominoEnv) ..	64
List Domino Release Information (QnninListDominoRlsI)	67
Get Notes.ini Value (QnninGetIniValue)	69
Get Notes.Ini Value - zero terminated (QnninGetIniValuez)	71
Get Server Document Item - zero terminated (QnninGetServerDocItemz)	72
Set Server Document Item (QnninSetServerDocItem)	72
Set Server Document Item - zero terminated (QnninSetServerDocItemz)	75

Chapter 1

Application Development on Domino for i5/OS

Welcome to the Application Development Guide for Domino for i5/OS. This document supplements the application development information that ships with Domino on all platforms by documenting Domino development differences and considerations unique to i5/OS. For more information on additional application development resources see, "Where to find information about Domino application development" later in this chapter.

About this guide

This guide covers the following information:

- Application Development Considerations for developing on i5/OS
- Considerations for using C
- Considerations for using C++
- Considerations for using Java
- Considerations for using LotusScript
- Integrating Notes and DB2 UDB for iSeries data
- Application Programming Interfaces (APIs) unique to i5/OS

System Requirements

To develop Domino applications you will need the following:

Workstation

- Microsoft Windows 2000 or Microsoft Windows XP
- 275 MB hard drive space
- 128 MB RAM (256 recommended)

Server requirements

- 60 MB hard drive space
- IBM iSeries ILE C Compiler - 5722-WDS Option 51
- OS/400 System Openness Includes - 5722SS1, option 13
- RUNDOMCMD - OS/400 Domino Command (ships with Domino 7 for i5/OS)

Note ASCII C/C++ Run Time for OS/400 is now shipped with the i5/OS platform, and is no longer required as a separate download. For more information, see

<http://www.ibm.com/servers/enable/site/asciirt/>

Additional resources for Domino application development

The following resources provide additional information for developing applications for Domino for i5/OS:

- Lotus developerWorks

The Lotus developerWorks Web site 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 on the Web at

<http://www.ibm.com/developerworks/lotus>

- Lotus Domino for i5/OS Developer's Tools Web site

The Domino for i5/OS Developer's Tools Web site contains tools and sample code that you can use to integrate Domino for i5/OS applications with i5/OS applications and data. The Web site is available at

<http://www.ibm.com/eserver/series/domino/domdevtools.html>

- The IBM eServer iSeries Information Center

The IBM eServer iSeries Information Center contains hardware, software and application development information in one resource. Domino for i5/OS 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 i5/OS integrated file system, including rules for specifying path names.

To access this information in the V5R1 version of the IBM eServer iSeries Information Center, click **Database and file systems->File systems and management**.

To access this information in the V5R2 version of the IBM eServer 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 V5R1 version of the IBM eServer iSeries Information Center, click **Database and file systems->DB2 UDB for iSeries**.

To access this information in the V5R2 version of the IBM eServer iSeries Information Center, click **Database**.

The Information Center ships on CD with every server and is available online at <http://publib.boulder.ibm.com/iseries/>

Chapter 2

Application development considerations

This chapter describes application development considerations unique to Domino for i5/OS.

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

The following requirements and restrictions apply to all programs that use the Notes C, C++, or Java API. For more information on any of these requirements and restrictions, see the appropriate sections later in this chapter.

- Obtaining the C, C++ and Java API.
- All programs must run under the QNOTES user profile.
- Be aware of threading considerations on i5/OS.
- 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 operators.
- As needed, change your programs to use the i5/OS 32-bit API handles.
- Make sure that your applications are enabled for teraspace.

Note While it has been recommended for several releases that customers teraspace-enable their Domino applications, the next major release of Domino may require that your application be enabled for teraspace. Service program extensions to Domino should also be enabled for teraspace.

- Make sure you properly convert between EBCDIC and LMBCS (ASCII).
- Program (*PGM) and service program (*SRVPGM) objects that bind to the LIBNOTES *SRVPGM must bind explicitly to LIBNOTES in the QNOTES library. Do not use the default value of *LIBL on the BNDSRVPGM keyword. If QNOTES is not specified as the library for LIBNOTES, then the wrong version of Domino may be used on multi-versioned systems.

Obtaining the C, C++ and Java API

Depending on the APIs you use to develop your Domino applications, you can obtain the Domino C, C++ and Java API 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 developerWorks Web site or the Lotus Domino for i5/OS Developer's Tools Website.
 - The Lotus developerWorks Web site is available at <http://www.ibm.com/developerworks/lotus>
 - The Lotus Domino for i5/OS Developer's Tools Web site is available at <http://www.ibm.com/eserver/iserics/domino/domdevtools.html>
- The Java API ships 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
 - QSYRLSPH - release profile handle

- Use the Submit Job command (SBMJOB) to submit the job to run under QNOTES.
- Use the RUNDOMCMD utility. 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 SBJJOB 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
      */

/* =====  =====
===== */

/* 10/21/2005 Clear      Update for multi-version support
      */

/* 02/18/2000 BEO      INITIAL INSTALL
      */

/*=====
=====*/

PGM

                                DCL      VAR(&SERVER) TYPE(*CHAR) LEN(10)

```

```

                                DCL          VAR(&PATH) TYPE(*CHAR) LEN(256)
                                DCL          VAR(&NOTESDATA) TYPE(*CHAR)
LEN(128)
                                DCL          VAR(&VERSION) TYPE(*CHAR) LEN(3)
                                DCL          VAR(&CLIB) TYPE(*CHAR) LEN(10)

/* TEMP */
                                CHGVAR     VAR(&SERVER) VALUE('as40019a')
                                CHGVAR     VAR(&NOTESDATA)
VALUE('/servers/16')
                                CHGVAR     VAR(&VERSION) VALUE('700')
                                CHGVAR     VAR(&CLIB) VALUE('QDOMINO' *TCAT
&VERSION)
/* ADD ENVIRONMENT VARIABLE PATH */
                                CHGVAR     VAR(&PATH) +
VALUE('/qibm/proddata/lotus/domino' *TCAT +
                                &VERSION *TCAT +
                                '/qibm/userdata/lotus/notes:'
*TCAT +
                                &NOTESDATA)
                                ADDENVVAR  ENVVAR(PATH) VALUE(&PATH)
                                MONMSG     MSGID(CPFA980) EXEC(CHGENVVAR
ENVVAR(PATH) +
                                VALUE(&PATH))

/* CHANGE CURRENT DIRECTORY */
                                CHGCURDIR  DIR(&NOTESDATA)

/* CHANGE CURRENT LIBRARY */
                                CHGCURLIB  CURLIB(&CLIB)

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

END:                            ENDPGM

```

Threading considerations for i5/OS

Consider the following when developing your Domino 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 i5/OS operating system has restrictions on jobs capable of running multiple threads and on the use of secondary threads. For details, see the Programming topic in the IBM eServer iSeries Information Center at

<http://publib.boulder.ibm.com/series/>

Certain Domino APIs require the ability to start another thread (the job it runs in must be multithread capable). Interactive jobs on i5/OS 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 IBM eServer iSeries Information Center at

<http://publib.boulder.ibm.com/series/>

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 currently cannot 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

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* API as described below 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:

- Use the QnninSetDomEnv API

Use this API to set the Domino server environment in your application. The QnninSetDominoEnv API is passed a parameter that has the Domino server's name. The API detects the Domino server's configured release and automatically sets the PATH environment variable correctly for that Domino server's configuration.

- Use the SETDOMENV CL command

Use the Set Domino Environment (SETDOMENV) CL command to set the Domino server environment before you run the application. The SETDOMENV command detects the Domino server's configured release and automatically sets the PATH environment variable correctly. This command may be used in an interactive job to establish the environment prior to submitting a batch job (SBMJOB) to run your Domino application. For example:

```
SETDOMENV SERVER(<myserver>) USER(*CURRENT)
```

```
SBMJOB CMD(CALL PGM(<mylib>/<mydomapp>)) JOB(<job-name>)  
USER(QNOTES) CPYENVVAR(*YES) ALWMLTTHD(*YES)
```

As a general rule, Domino API programs may fail or hang if called in an interactive job because interactive jobs are not multi-thread capable.

- Launch your program using the RUNDOMCMD CL Command which sets the PATH

Use the Run Domino Command (RUNDOMCMD) CL command to launch your stand-alone application against a particular Domino server. You can submit this command to run in batch and it runs with the correct Domino server environment, under the correct user profile of QNOTES, etc. The Domino server does not need to be running for this command to work. The RUNDOMCMD command detects the Domino server's configured release and automatically sets the PATH environment variable correctly for that server's configuration.

Which of these techniques to use depends on whether you need to set the Domino server environment from within a C/C++ application, or if you set the Domino server environment from a CL program or interactively before you run an application. If you need to set the Domino server environment from within a C/C++ application, you would probably use one of the Qnnin* APIs. If you set the Domino server environment from a CL program or interactively before you run an application, you would most likely use a CL command.

It is important that you do not hard code the /QIBM/ProdData/LOTUS/NOTES directory and assume that it is correct for all Domino servers. You must consider this directory as something that can be different for each Domino server, just like the data directory for each Domino server is different.

Refer to Chapter 4, Domino for i5/OS server APIs for more information on using the QnninSetDomEnv API.

Considerations for multi-versioning

Note For more information and considerations for programming in a multi-version Domino environment refer to the Redbook *Lotus Domino 6 Multi-Versioning Support on the IBM eServer iSeries Server*.

With non multi-version capable Domino releases, the /QIBM/ProdData/LOTUS/NOTES directory is the Domino product executable directory for every release of Domino. Users and applications should not place anything into this directory or use it directly. Existing applications may have referenced this executable directory when setting the PATH environment variable if the application can run Domino applications outside of a Domino server's context.

With the support for multiple Domino releases on a single system or partition, each multi-version Domino release has a different ProdData directory. This directory is /QIBM/ProdData/LOTUS/DOMINOXXX, where XXX is a number corresponding to the release. For example, the ProdData directory for Domino 6.5.1 is /QIBM/ProdData/LOTUS/DOMINO651.

With multi-version capable releases like Domino 7.0 and Domino 6.5.1, the /QIBM/ProdData/Lotus/Notes directory still exists, but is now a symbolic link to the primary release of Domino (the primary release is the most recent release of Domino installed on the system or partition). Applications using /QIBM/ProdData/LOTUS/NOTES in their PATH will continue to work, but only when one release of Domino is installed.

If the system has multiple releases of Domino installed, the environment for these applications must have the correct Domino product executable directory in their PATH based on the Domino server against which they are running. Applications that use /QIBM/ProdData/LOTUS/NOTES when setting the PATH environment variable fail when the application tries to reference a Domino database that is on any Domino server that is not configured to use the primary release. Applications that reference /QIBM/ProdData/LOTUS/NOTES should be updated so that the correct multi-version information is set in the PATH environment variable regardless of the Domino server's configured release.

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 *Integrated File System* topic in the IBM eServer iSeries Information Center at

<http://publib.boulder.ibm.com/iseries/>

Restriction on APIs in exception or cancel handlers

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

If a Domino API call is invoked in your exception or cancel handler, a deadlock could occur. If a Domino API call in your application is holding a lock when an exception or cancel occurs and the handler issues another Domino API call, 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 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 i5/OS restriction that there can be only one global `new()` and `delete()` operator per activation group.

Using API handles

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

Enabling your application for teraspace

"Teraspace" is the term for process-local storage on i5/OS. 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 (SLS) data. Domino for i5/OS makes use of teraspace in certain areas where it needs more than 16MB of contiguous storage. Domino is also fully enabled for teraspace; you can pass teraspace memory to Domino and it can access the larger memory allocations.

While it has been recommended for several releases that customers enable their Domino applications for teraspace, the next major release of Domino may require that your application be enabled for teraspace. Service program extensions to Domino should also be enabled for teraspace.

When creating modules, programs or service programs, follow these guidelines to ensure that your Domino applications are enabled for teraspace and are able to run in this environment:

- All programs and service programs that use the Domino C/C++ API should be enabled for teraspace.
- Modules created using the CRTCMOD or CRTCPPMOD commands are enabled for teraspace by specifying the TERASPACE(*YES) keyword option. In addition, use STGMDL(*INHERIT), which provides the most flexibility when using these modules in programs and service programs later.
- Programs created using the CRTPGM command are enabled for teraspace if the modules are also enabled for teraspace. Specify TERASPACE(*YES) and STGMDL(*SINGLVL) on the CRTBNDC command if you are creating a module and program in one step.
- When you create a bound service program using the CRTSRVPGM command, specify ACTGRP(*CALLER) and STGMDL(*INHERIT) for a neutral storage model. This will create a service program that will be compatible with current and future releases of Domino.

Note For more information on enabling your application for teraspace, see Chapter 4 of the *iSeries ILE Concepts (SC41-5606)* book, which is available at

<http://publib.boulder.ibm.com/series/v5r2/ic2924/books/c4156066.pdf>

See also the Domino for i5/OS C and C++ toolkits, which can be downloaded from Lotus developerWorks at

<http://www.ibm.com/developerworks/lotus>

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

The ASCII C/C++ Run Time provides a seamless solution for conversion between an ASCII application and the EBCDIC ILE environment of Domino for i5/OS. 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 server.

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

ASCII C/C++ Run Time ships with IBM i5/OS.

Where to find the header files on i5/OS

When you install the application development software, the files are loaded into specific i5/OS libraries. The following table shows the i5/OS library in which the tools support is installed and the path for accessing that library through the integrated file system.

<i>Item</i>	<i>Library</i>	<i>Path</i>
C APIs	QNOTESAPI	/qsys.lib/qnotesapi.lib
C++ APIs	QNOTESCPP	/qsys.lib/qnotescpp.lib
LotusScript Extensions	QNOTESLSKT	/qsys.lib/qnoteslskt.lib

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 and linking C source on i5/OS

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 i5/OS C compiler. 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, such as `#include "inc_name"`, to IFS files, such as `[hpp_search_path]/inc_name`, where the `hpp_search_path` is defined by the compiler option `INCDIR`. You must ensure the C headers are located as IFS files, using one of the following two procedures:

- 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. 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 i5/OS, 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 i5/OS 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.

The following 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/<mylib>.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 LotusScript, simply code runthis("parms_go_here") where "parms_go_here" are the parameters being used. For example:

```
cmdstring="input string"

rc=runthis(cmdstring)

Print "Cmdstring: ";cmdstring
```

The C service program would look like the following:

```
#include <stdlib.h>
/* This simple service program function changes the string
passed in on cmd to the text 'done
calling' */
int runthis(char *cmd)
{
int rval=0;
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 (help7_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:

```
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 i5/OS, the C header files are installed in the QNOTESAPI library as members of the i5/OS file H.FILE. The installation also creates symbolic links in the i5/OS 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 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 and linking C source on i5/OS

You can compile and link C code with source in either an i5/OS physical file or an IFS file. Detailed information on these procedures can be found in the *WebSphere Development Studio ILE C/C++ Programmer's Guide*. Refer to "Enabling your application for Teraspace" for additional instructions on how to create modules, programs and service programs that are enabled for teraspace.

Compiling and linking using a source physical file

Follow these steps to compile and link source in an i5/OS physical file.

1. Put the QNOTESAPI library in your library list. For example, enter the following i5/OS command:

```
addliblible qnotesapi
```

2. Specify a define name of OS400 when you create the C module using the CRTCMOD command. Remember to specify the options necessary to enable the module for teraspace.

```
crtcmmod module(ctest/foo) define(os400) teraspace(*yes)
stgmdl(*inherit)
```

This command creates a module called FOO that is enabled for teraspace in the library CTEST.

3. After compiling the C module, link the module into a program (CRTPGM command) or a service program (CRTSRVPGM command). When you use either the CRTPGM or the CRTSRVPGM command, bind in the LIBNOTES service program in the QNOTES library. For example:

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

This command creates a program called FOO in the library CTEST. This program is also enabled for teraspace.

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)
bndsrvpgm(qnotes/libnotes)
```

4. After the program or service program is created, remember to change the ownership of the object to the QNOTES user profile. The QNOTES user profile must also be authorized to access the library that contains the program or service program if the application will run under the Domino server. QNOTES can be explicitly authorized to the library, or the library may allow *PUBLIC access. For example:

```
chgobjown obj(ctest/foo) objtype(*pgm) newown(qnotes)

grtobjaut obj(qsys/ctest) objtype(*lib) user(qnotes)
aut(*use)
```

5. 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. For example:

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

You can run the program by using the i5/OS CALL command. The program must run under the QNOTES user profile and you must set up a PATH environment variable to identify the required i5/OS directories. You can swap to the QNOTES user profile in your program or you can specify QNOTES when you run the program.

The i5/OS Submit Job (SBMJOB) command allows you to specify a user profile when you run a program. Here is an example of using the SBJOB 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 using an IFS file

Follow these steps to compile and link source in an i5/OS IFS file.

The commands below use the /QIBM/PRODDATA/QADRT/INCLUDE directory and QADRT/QADRTNTS service program which are included with the ASCII C/C++ Run Time.

1. Compile the Domino C application source in IFS file /ntcapi/csample/csample.c. For example:

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

This command compiles the CSAMPLE source code using includes in the QADRT and NOTESAPI directories and creates a module object in the NTCAPI library. This module is also enabled for teraspace.

2. After compiling the C module, link the module into a program (CRTPGM command) or a service program (CRTSRVPGM command) and bind in the LIBNOTES service program in the QNOTES library. For example:

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

If you create a service program, remember to specify the recommended values for enabling teraspace. For example:

```
CRTSRVPGM SRVPGM (NTCAPI/CSAMPLE) MODULE (NTCAPI/CSAMPLE)
BNSRVPGM (QNOTES/LIBNOTES QADRT/QADRTNTS)
CTGRP (*CALLER) STGMDL (*INHERIT)
```

3. After the program or service program is created, remember to change the ownership of the object to the QNOTES user profile. The QNOTES user profile must also be authorized to access the library that contains the program or service program if the application will run under the Domino server. QNOTES can be explicitly authorized to the library, or the library may allow *PUBLIC access. For example:

```
CHGOBJOWN OBJ (NTCAPI/CSAMPLE) OBJTYPE (*PGM) NEWOWN (QNOTES)
GRTOBJAUT OBJ (QSYS/NTCAPI) OBJTYPE (*LIB) USER (QNOTES)
AUT (*USE)
```

4. If the application runs under the Domino server, create a symbolic link from the Domino server Userdata directory to the program or service program. For example:

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

Considerations for developing applications that use Java

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

- i5/OS software required to run Java programs or agents
- Setup required to run Java programs
- Performance when using Java in Domino Agents on i5/OS.
- Restrictions and requirements for running Java agents
- Requirements for compiling Java programs, agents, and servlets
- JavaMaxHeapSize
- JavaOS400CHKPATH

i5/OS software requirements for Java

To run Java programs or Java agents on i5/OS, you must install the following software:

- OS/400 - Qshell Interpreter, 5722SS1 option 30
- Java Developer Kit 1.4, 5722JV1 option 6

Note Domino 6 for i5/OS requires Java Developer Kit 1.3, 5722JV1 option 5.

Domino for i5/OS uses the Java Virtual Machine that ships with i5/OS.

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

Note If there is more than one release on the system, this value should be set to /QIBM/ProdData/Lotus/Domino.XXX/Notes.jar, where XXX is the number corresponding to the release you are working with (ex. 700 for Domino 7.0).

- The directory that contains the class files for the Java program
- Add QNOTES to the library list:

```
addlibl qnotes
```

When you run the Java program, submit it under the QNOTES user profile. For example, to run the Java program MyProg, enter the following i5/OS 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 13 of *Installing and Managing Domino 7 for i5/OS* (i400help.pdf).

Using Java in Domino Agents on i5/OS

When you use Java in a Domino agent on i5/OS, 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 i5/OS 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 `note.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/Lotus/DOMINOXXX/data/domino/
java/ncso.jar
```

where *XXX* is the number corresponding to the release of Domino (ex. 700 for Domino 7.0.0).

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 i5/OS, a .class or .jar file will load and run 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` of 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 i5/OS applications, consider the following:

- When you install the Java Developer Kit on your 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 cannot load or run native methods that call back into Domino.
- If Java Web Agents are enabled, it is critical that the Web agent timeout field in the server document is set to ZERO or some high value that will never be hit. If Java Agents time out on the server, problems can occur.

Requirements for compiling Java programs, agents, and servlets

You can compile your Java programs, agents, and servlets on any platform that has the Java 1.4 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 i5/OS directory:

```
/QIBM/ProdData/Lotus/DOMINOXXX
```

where XXX corresponds to the release of Domino (ex. 700 for Domino 7.0.0).

For more information about Java and Domino, see the Domino Application development documentation that ships with the Domino Designer application or the Lotus developerWorks Web site at

<http://www.ibm.com/developerworks/lotus>

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 1GB 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.

Note 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 beneficial to set the JavaMaxHeapSize to *NOMAX.

JavaOS400CHKPATH

CHKPATH is a variable that can be set when running any i5/OS Java application, Domino servlets and Domino agents. When using RUNJAVA on i5/OS, 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 warning 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 joblog.

Considerations for LotusScript functions on i5/OS

Various LotusScript functions are not supported or work differently because of the integrated storage management and operating system 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 ("").
- **Date, Date\$**
Changing the date 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 i5/OS 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. i5/OS 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 i5/OS 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 i5/OS.
 - 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 i5/OS through LotusScript is not supported. Generates a run-time error.

File system differences

- There are no drive letters on i5/OS. If you use a path name containing a drive letter, LotusScript may return an error.
- i5/OS 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 i5/OS or UNIX that uses a backslash in a path name string literal will not work.
 - 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 i5/OS.
- 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 i5/OS. This difference affects the CreateObject, GetObject, IsObject, and IsUnknown functions. The CreateObject and IsObject functions will raise run-time errors when run on i5/OS. 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 i5/OS. The IsUnknown function always returns FALSE 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 i5/OS 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 i5/OS 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 i5/OS files. Similarly, LotusScript statements that read from a file (such as Get, Input #, and Line Input #) will read i5/OS files as EBCDIC characters.

You can convert the EBCDIC file to ASCII by using the i5/OS 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) ")
```

Using the LotusScript Declare Function statement to call C functions

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 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("parms_go_here") where "parms_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.

Note Domino runs internally in Lotus Multi-Byte Character Set (LMBCS), and the character string representation for the C API parameters is assumed to be LMBCS. When calling Domino C API and passing string type parameters in LotusScript, parameters must be declared as LMBCS types. For example:

```
Declare Function NSFDbCreateAndCopy Lib LibName Alias
"NSFDbCreateAndCopy" (_
Byval srcDb As Lmbcs String , _
Byval dstDb As Lmbcs String, _
Byval noteClass As Integer, _
Byval limit As Integer, _
Byval flags As Long, _
hDb As Long) As Integer
```

If LMBCS is not specified in this way, the default is to pass all strings in a type argument in the platform-native character set, which for i5/OS would be EBCDIC.

For parameters other than pointers, you should indicate `_System` linkage in your routine. This is only supported by the i5/OS ILE C compiler. Here is an example of the `_System` linkage statement:

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

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 run.

`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.

About adding hook drivers

On i5/OS, 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('qsys.lib/mylib.lib/libhook.srvpgm')
newlnk('qibm/userdata/lotus/dominoXXX/libhook.srvpgm')
```

where `XXX` is the number corresponding to the release of Domino (ex. 700 for Domino 7.0.0).

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 i5/OS is the integration between Domino and DB2 Universal Database for iSeries databases. You can use the following methods to access the DB2 UDB 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 i5/OS, 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 i5/OS 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 Integrator™ (LEI) 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 IBM eServer 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 i5/OS 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 i5/OS command line, enter the following command:
`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 i5/OS command line, enter the following command:
`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 i5/OS. 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 LotusScript data object (LS:DO) 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 i5/OS user profile and password. Before allowing the connection, i5/OS 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 i5/OS 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 i5/OS 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 i5/OS user profile" describes how your Domino application can provide the i5/OS user profile and password.
 - Set up special i5/OS user profiles whose only function is to provide Domino access to i5/OS data. This eliminates the need for each Domino user to have an i5/OS user profile. "Connecting a Domino application to DB2 UDB for iSeries with a special i5/OS 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 i5/OS 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 i5/OS 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 i5/OS 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 i5/OS 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 i5/OS user profile and password.

When you want your Domino application to connect using the Domino user's i5/OS profile, do one of the following:

- **Prompt the user:** Your application can prompt the Domino user for an i5/OS 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 i5/OS user profiles on your server.
- **Store the user ID and password:** You can provide a form and database for your Domino users who need database access. The form would prompt the user for the i5/OS 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 i5/OS 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 i5/OS password.

Connecting a Domino application to DB2 UDB for iSeries with a special i5/OS 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 i5/OS 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 i5/OS 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, i5/OS 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 i5/OS 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 i5/OS 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 server, 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 i5/OS 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 i5/OS data, be aware of the following:

- For LS:DO, you must specify the following "UseLSX" statement in the Event (Options) within 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 following SQL naming convention:

```
collection.table
```

Do not use:

```
collection/table
```

- For i5/OS, 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 i5/OS by using the Work with Relational Database Entries (WRKRDBDIRE) command to add the data source to the i5/OS relational database directory.

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

<http://publib.boulder.ibm.com/iseries/>

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 i5/OS prestarted jobs for processing SQL requests. These jobs are started when the Domino server is started on i5/OS 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 the following i5/OS 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 QSQSRVR 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 i5/OS, 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:

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

Instead, use a SQL statement such as the following:

```
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:

```
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:

<i>Value</i>	<i>Format</i>	<i>Example</i>
Date	"yyyy-mm-dd"	"1997-07-31"
Time	"hh:mm:ss"	"12:15:30"
Time Stamp	"yyyy-mm-dd-hh.mm.ss.mmmmmm"	"1997-07-31-12.15.35.000000"

- 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 i5/OS:
 - 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:

<u>SQL data type</u>	<u>DB_INFO_LENGTH indicates</u>
integer, small integer, float	number of bytes
char, varchar, longvarchar	number of characters
numeric, decimal	(not applicable)

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 run. 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 server 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 server.

First, sign onto the remote server and perform the following operations:

1. Enter the command:

```
wrkobj obj(qgpl/*all) objtype(*sqlpkg)
```
2. Look for these two objects:
 - QSQCLIPKGC
 - QSQCLIPKGN
3. If the object QSQCLIPKGC exists, delete it:

```
dltsqlpkg sqlpkg(qgpl/qsqclipkgc)
```

4. If the object QSQCLIPKGN exists, delete it:

```
dltssqlpkg sqlpkg(qgp1/qsqclipkgn)
```

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

Event Options:

```
Option Public
Uselsx "*lsxodbc"
```

Event Initialize:

```
Sub Initialize
    Dim con As New ODBCConnection
    con.AutoCommit=False
    If (con.ConnectTo("DataSource", "UserID", "Password")) Then
        Call con.Disconnect
    End If
End Sub
```

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 i5/OS 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 i5/OS APIs

This chapter describes APIs unique to Domino for i5/OS.

Domino for i5/OS 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 job's 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 job's Domino server environment.
- **List Domino Release Information (QnninListDominoRlsI)** is used to retrieve a list of installed Domino releases on the current 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 API programs at

<http://www.ibm.com/eserver/series/domino/domdevtools.html>

These API functions are included in the base option of Domino for i5/OS, 5733LD7. If 5722SS1 option 13 (OS/400 System Openness Includes) has already been installed when Domino 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. 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:

```
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:

1	Data buffer for Domino servers	O	Char(*)
2	Data buffer length	I	Binary(4)
3	Format name	I	Char(8)
4	Error Code	I/O	Char(*)

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 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.

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

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:

1	Data buffer for Domino server information	O	Char(*)
2	Data buffer length	I	Binary(4)
3	Server name	I	Char(255)
2	Server name length	I	Binary(4)
4	Error Code	I/O	Char(*)

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.

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

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

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:

1	Data buffer for Domino server information	O	Char(*)
2	Data buffer length	I	Binary(4)
3	Server name	I	Char(255)
4	Server name length	I	Binary(4)
5	Format name	I	Char(8)
6	Error Code	I/O	Char(*)

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

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

DATR0200 Attribute Buffer

<i>Offset Dec</i>	<i>Offset Hex</i>	<i>Type</i>	<i>Field</i>
0	0	DATR0100	DATR0100 Buffer
		Buffer (see above)	
80	50	Character(1)	Log replication events
81	51	Character(1)	Log client session events
82	52	Character(1)	DOLS enabled
83	53	Character(1)	Day light savings time enabled
84	54	Character(6)	Time zone
90	5A	Character(10)	Collation sort order
100	64	Character(11)	SMTP type
111	6F	Character(1)	Directory type
112	70	Character(1)	Application Server Provider
113	71	Character(3)	Reserved
116	74	Binary(4)	Offset to organization name
120	78	Binary(4)	Length of organization name
124	7C	Binary(4)	Offset to web browser list, List Type 1
128	80	Binary(4)	Offset to News Readers List List Type 1
132	84	Binary(4)	Offset to Mail services List List Type 1
136	88	Binary(4)	Offset to Advanced services List List Type 1
140	8C	Binary(4)	Offset to Connection Services List List Type 1

continued

<i>Offset Dec</i>	<i>Offset Hex</i>	<i>Type</i>	<i>Field</i>
144	90	Binary(4)	Offset to Directory Services List List Type 1
148	94	Binary(4)	Offset to Additional Services List List Type 1
152	98	Binary(4)	Offset to TCP Info List List Type 2
156	9C	Binary(4)	Offset to canonical name
160	A0	Binary(4)	Length of canonical name
164	A4	Binary(4)	Offset to server title
168	A8	Binary(4)	Length of server title
172	AC	Binary(4)	Offset to server host name
176	B0	Binary(4)	Length of server host name
180	B4	Binary(4)	Offset to domain name
184	B8	Binary(4)	Length of domain name
188	BC	Binary(4)	Offset to administrator(s) name(s)
192	C0	Binary(4)	Length of administrator(s) name(s)
*	*	Character(*)	Variable data

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:

*SAME	HU - Hungarian	RU - Cyrillic
*STD - Standard	IS - Icelandic	SK - Slovak
CS - Czech	IT - Italian	SL-SI - Slovenian
DA-DK-AA - Nordic 2to1	JA - Japanese	SQ-AL- Albanian
DE- German	KO- Korean	SV - Swedish
E2-ES - Spanish - Modern	LT - Lithuanian	TH - Thai
EL- Greek	LV - Baltic	TR-Turkish
EN - Latin1	NL- Dutch	ZH-CN - Simplified Chinese
ES - Spanish - Traditional	NO- Nordic	ZH-TW Traditional Chinese
ET-EE - Estonian collation	PL - Latin2	FI - Finnish
PL-PL - Polish	FR - French	PT - Portuguese
HR- Croatian	RO - Romanian	

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:

GMT - Greenwich Mean Time	BST - Bering Standard Time	HST - Alaska-Hawaii Standard Time
ZW1 - 1 hour West of GMT	ZW12 - 12 hours West of GMT	ZE6B - 6 1/2 hours East of GMT
ZW2 - 2 hours West of GMT	ZE12C - 12 3/4 hours East of GMT	ZE6 - 6 hours East of GMT
ZW3 - 3 hours West of GMT	ZE12 - 12 hours East of GMT	ZE5C - 5 3/4 hours East of GMT
NST - Newfoundland Standard Time	ZE11B - 11 1/2 hours East of GMT	ZE5B - 5 1/2 hours East of GMT
AST - Atlantic Standard Time	ZE11 - 11 hours East of GMT	ZE5 - 0 hours East of GMT
EST - Eastern Standard Time	ZE10B - 10 1/2 hours East of GMT	ZE4B - 4 1/2 hours East of GMT
CST - Central Standard Time	ZE10 - 10 hours East of GMT	ZE4 - 0 hours East of GMT
MST - Mountain Standard Time	ZE9B - 9 1/2 hours East of GMT	ZE3B - 3 1/2 hours East of GMT
PST - Pacific Standard Time	ZE9 - 9 hours East of GMT	ZE3 - 0 hours East of GMT
YST - Yukon Standard Time	ZE8 - 8 hours East of GMT	ZE2 - 0 hours East of GMT
ZW9B - 9 1/2 hours West of GMT	ZE7 - 7 hours East of GMT	CET - Central European Time

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

<i>Offset Dec</i>	<i>Offset Hex</i>	<i>Type</i>	<i>Field</i>
0	0	Binary(4)	Number of Items in List
4	4	Binary(4)	Length of each item
*	*	Character(*)	Items in the list

List Type 2

<i>Offset Dec</i>	<i>Offset Hex</i>	<i>Type</i>	<i>Field</i>
0	0	Binary(4)	Number of Items in List
*	*	Character(*)	TCP/IP list items in TCP List Format

TCP List Format

<i>Offset Dec</i>	<i>Offset Hex</i>	<i>Type</i>	<i>Field</i>
0	0	Binary(4)	OffsetToNext Item
4	4	Character(17)	TCP/IP Address
21	15	Character(1)	Port Enabled
22	16	Character(10)	Port Encrypted
32	20	Binary(4)	Offset To Port Name
36	24	Binary(4)	Length of Port Name
40	28	Binary(4)	Offset To Network Name
44	2C	Binary(4)	Length of Network Name
48	30	Character(1)	Port Compression
49	31	Character(3)	Reserved
52	34	Binary(4)	Offset To IP host name
56	38	Binary(4)	Length of IP host name
*	*	Character(*)	Variable data

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

TCPIP 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:

1	Action to Perform	I	Binary(4)
2	Server name	I	Char(255)
3	Server name length	I	Binary(4)
4	User Profile Action	I	Binary(4)
5	Path Action	I	Binary(4)
6	Error Code	I/O	Char(*)

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 job's 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:

1	Data buffer for Domino environment	O	Char(*)
2	Data buffer length	I	Binary(4)
3	Format name	I	Char(8)
4	Error Code	I/O	Char(*)

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

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

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 system.

Required Parameter Group:

1	Data buffer for Domino releases	O	Char(*)
2	Data buffer length	I	Binary(4)
3	Format name	I	Char(8)
4	Error Code	I/O	Char(*)

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.

<i>Offset Dec</i>	<i>Offset Hex</i>	<i>Type</i>	<i>Field</i>
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Binary(4)	Offset to release entries
12	C	Binary(4)	Number of release entries
16	10	Binary(4)	Length of each release entry

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.

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

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:

1	Server name	I	Char(255)
2	Server name length	I	Binary(4)
3	Notes.ini variable name	I	Char(*)
4	Notes.ini variable name length	I	Binary(4)
5	Return Buffer	O	Char(*)
6	Return Buffer Length	I	Binary(4)
7	Return Bytes Available	O	Binary(*)
8	Error Code	I/O	Char(*)

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:

1	Server name	I	Char(255)
2	Notes.ini variable name	I	Char(*)
3	Return Buffer	O	Char(*)
4	Return Buffer Length	I	Binary(4)
5	Return Bytes Available	O	Binary(*)
6	Error Code	I/O	Char(*)

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.

Required Parameter Group

1	Server name	I	Char(255)
2	Server document item name	I	Char(*)
3	Return Buffer	O	Char(*)
4	Return Buffer Length	I	Binary(4)
5	Return Bytes Available	O	Binary(*)
6	Return Buffer data type	I	Binary(4)
7	Error Code	I/O	Char(*)

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:

1	Server name	I	Char(255)
2	Server name length	I	Binary(4)
3	Server document item name	I	Char(*)
4	Server document item name length	I	Binary(4)
5	New Value	I	Char(*)
6	New Value Length	I	Binary(4)
7	List Processing Flags	I	Binary(4)
8	New Value data type	I	Binary(4)
9	Error Code	I/O	Char(*)

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:

0	Replace	Replace the value that may currently be set.
1	Append	Append the new value to an existing list.
2	Remove	Remove the value from a list. The value in New Value will be removed from the list associated with the item name.

New Value data type

INPUT; BINARY(4)

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

1	Text Data	New Value is a CCSID 37 CHAR(*)
2	Float Data	New Value is a C/C++ compatible "double" data type.
3	Integer Data	New Value is a BINARY(4) value.

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

1	Server name	I	Char(255)
2	Server document item name	I	Char(*)
3	New Value	I	Char(*)
4	List Processing Flags	I	Binary(4)
5	New Value data type	I	Binary(4)
6	Error Code	I/O	Char(*)

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

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.

Index

A

- Agent, 16, 22
 - calling program, 16, 29
 - Java, 22, 23
- API, 6, 45, 46
 - 32-bit handles, 13
 - ASCII and EBCDIC, 14
 - C considerations, 15
 - exception or cancel handlers, 12
 - global operators, 13
 - location of header files, 15
 - PATH environment variable, 10
 - server add-ins, 12
 - thread safety, 9
- APPDEV.NSF, 18
- Application development
 - API handles, 13
 - ASCII and EBCDIC, 14
 - C considerations, 15
 - calling programs, 29
 - exception or cancel handlers, 12
 - general considerations, 5
 - global operators, 13
 - location of header files, 15
 - LotusScript considerations, 26
 - PATH environment variable, 10
 - server add-ins, 12
 - thread safety, 9
- ASCII, 14, 15, 17

C

- C API, 6, 16
 - accessing header files, 18
 - ASCII compilation, 17
 - ASCII-to-EBCDIC conversion, 14
 - compiler requirements, 16
 - compiling on AS/400, 18
 - general considerations, 5
- C++ API, 6
 - EBCDIC requirement, 14
 - general considerations, 5
- Cancel handler, 12
- CHKPATH, 24
- CLASSPATH environment variable, 21

- Command
 - ADDENVVAR, 10
 - ADDLIB, 18, 21
 - ADDLNK, 12, 18
 - CPYTOSTMF, 18
 - SBMJOB, 21
 - WRKRDBDIRE, 34
- Compiler
 - C API, 16
- Connection services, 33

D

- Database integration
 - @Db functions, 38
 - errors messages, 43
 - LS:DO, 38, 39
 - methods, 33
 - relational database entry, 34
 - remote connections, 42
- DB2 UDB for iSeries, 34
 - ways of accessing, 33
- DB2/UDB for iSeries, 35
- DECS, 33
- Drive
 - shared, 18
- Environment variable
 - adding, 10
 - CLASSPATH, 21
 - PATH, 10, 21
- Examples
 - setting environment variable, 10
- Exception handler, 12

F

- FTP, 18

G

- Get Domino Environment, 64
- Get Notes.ini Value, 69
- Get Notes.Ini Value - zero terminated, 71

- Get Server Document Item - zero terminated, 72

H

- Header files
 - accessing from compiler, 18
 - where to find, 15
- Hook driver, 31

J

- Java, 6, 22, 24
 - AS/400 software required, 21
 - calling program, 29
 - general considerations, 5
 - restrictions for running agents, 23
 - setting up to run, 21
- JavaMaxHeapSize, 24

L

- LIBASCII, 15
- Library
 - QNOTESAPI, 15
 - QNOTESCPP, 15
 - QNOTESLSKT, 15
- List Domino Release Information, 67
- List Domino Servers, 46
- LMBCS, 14, 17
- LotusScript
 - considerations, 26
- LS:DO
 - calling program, 29
 - differences, 39
 - error messages, 43
 - performance, 38
 - prestarted jobs, 39
 - remote connections, 42
 - requirements, 38

M

- Message
 - database integration, 43

N

NotesPump, 34

P

PATH environment variable, 10, 21

Prestarted job, 39

Q

QnninGetDominoEnv, 45, 62, 64

QnninGetIniValue, 45, 69

QnninGetIniValuez, 45, 71

QnninGetServerDocItem, 45

QnninGetServerDocItemz, 45

QnninListDominoRlsI, 45, 67

QnninListDominoServers, 45, 46

QnninRtvDominoServerAttr, 45, 52

QnninRtvDominoServerI, 45, 49

QnninSetDominoEnv, 45

QnninSetIniValue, 45

QnninSetIniValuez, 45

QnninSetServerDocItem, 45, 72

QnninSetServerDocItemz, 45, 72, 75

QNOTES user profile

running application, 6

QSYS.LIB file system

location of header files, 15

R

Relational database name, 34

Remote connection, 42

Restrictions

exception or cancel handlers, 12

global operators, 13

Java agents, 23

LS:DO, 39

thread safety, 9

Retrieve Domino Server Attributes, 52

Retrieve Domino Server Information, 49

S

Server Information, 45

Set Domino Environment, 62

Set Server Document Item, 72

Set Server Document Item - zero

terminated, 75

Setup, 2

Java, 21

SQL, 39

Symbolic link, 12, 31

T

Teraspace, 13

Thread safety, 9

U

User profile

QNOTES, 6

Z

Zip file, 18