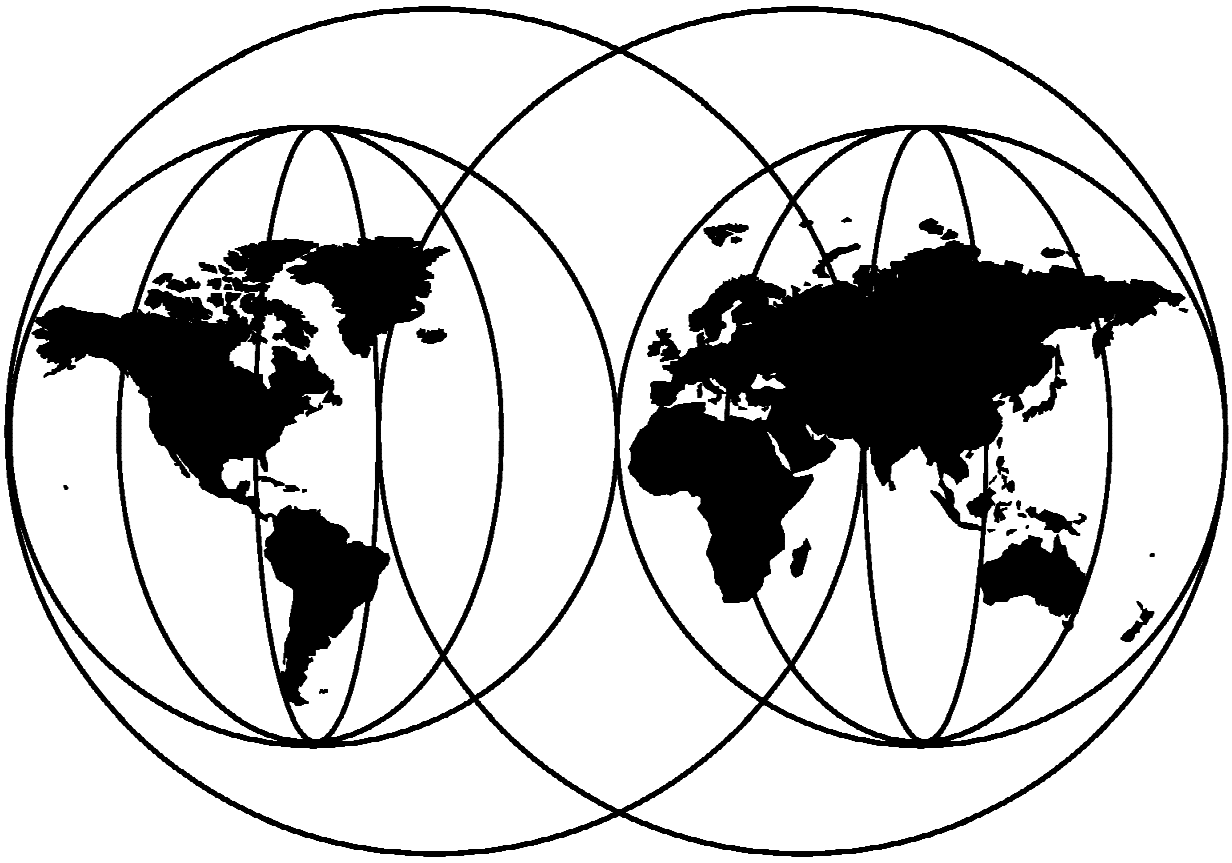
IBM

# Porting C Applications to Lotus Domino on S/390

*Alain Atge* *John LaFata* *Brian Macfaden*



**International Technical Support Organization**

http://www.redbooks.ibm.com

SG24-2092-00

International Technical Support Organization

SG24-2092-00

# Porting C Applications to Lotus Domino on S/390

February 1998

**Take Note!**

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special Notices” on page 63.

**First Edition (February 1998)**

This edition applies to the Lotus Notes C API Toolkit Release 4.51 for use with OS/390.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

# Contents

# Figures

# Tables

# Preface

This redbook will help you port C language applications that interface to Lotus Notes to the Lotus Domino server for S/390.

We worked with Application Partners, Inc, a software vendor, to port their WIT application from OS/2 and Windows NT to S/390. This redbook documents our experiences.

We used the Notes C Application Programming Interface (API) Toolkit that is provided by Lotus development to allow you to write C applications that access Notes databases. This toolkit is available on S/390.

We also used the libascii package available from IBM to assist with ASCII-to-EBCDIC data conversion on the S/390 server.

## The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

**Alain Atge** is a consultant in solution test and application porting. He works in IBM's porting center in Montpellier, France helping software vendors port their applications to S/390. Alan has 18 years of experience in application architecture, design and development. He worked for 10 years in customer technical support. His areas of expertise include programming languages such as Assembler, C and C + + and the OS/2 and Windows NT environments.

**John LaFata** is a lead programmer at Application Partners Inc. He maintains and develops the WIT product. John has extensive experience in C and the Lotus Notes C API toolkit. He also knows UNIX. He was involved in the port of the WIT product from OS/2 to Windows NT.

**Brian Macfaden** is a program manager at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on the value of the IBM System/390 platform and on the Lotus Domino Server on S/390. Brian has 20 years of experience in the information technology industry, specializing in networking, distributed systems and large systems. Before joining the ITSO he worked in Manchester, England.

Thanks to the following people for their invaluable contributions to this project:

| | |
|---|---|
| Richard Wilson | Application Partners Inc. |
| Peter Oliver | IBM S/390 Division |
| Gary Sutherland | IBM S/390 Division |
| Joe Sweeney | IBM S/390 Division |
| John Woods | IBM S/390 Division |
| Bob Haimowitz | ITSO, Poughkeepsie Center |
| Rich Conway | ITSO, Poughkeepsie Center |

## Comments Welcome

**Your comments are important to us!**

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 75 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

# Chapter 1. Summary

This book was written as the result of a project to port a Notes C application to Lotus Domino for S/390.

## 1.1 Most Notes Applications Port with Just a Replication

Most Notes applications require almost no effort to migrate them to S/390. Applications written using commands, @functions and LotusScript reside entirely within a Notes database. To migrate those applications to the Domino server on S/390, all you need to do is replicate or copy the database to OS/390. The code then runs in the Domino server environment exactly the same as on any other Domino server. This is shown in Figure 1.

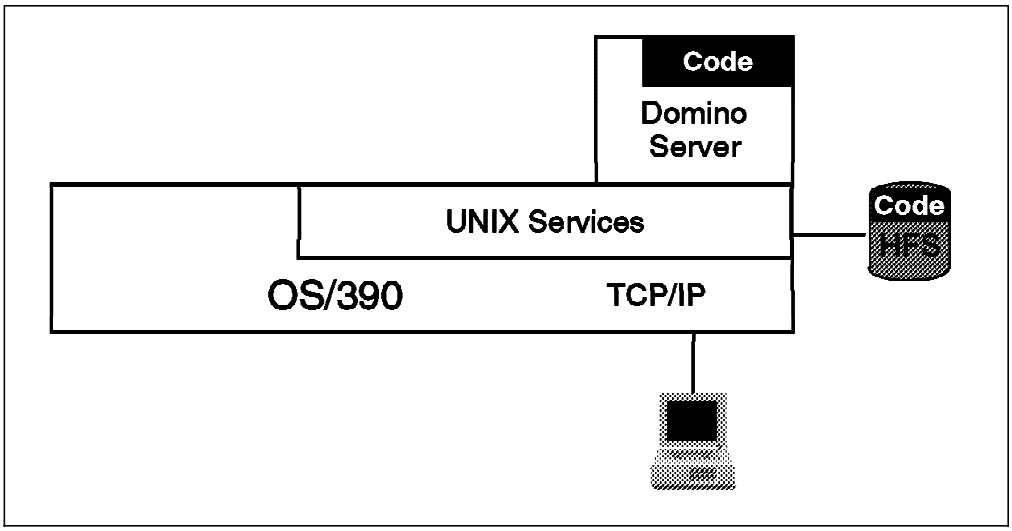



*Figure 1. Porting an Application in a Notes Database*

Notes databases are encoded in Lotus Multi-Byte Character Set (LMBCS) which uses an ASCII encoding method. We store Notes databases on S/390 in LMBCS also. (S/390 is an EBCDIC platform, but it can store ASCII files.) Therefore the database on S/390 is exactly the same as on any other platform. You can copy the database to the S/390 server using Notes replication or using utilities such as file transfer program (ftp) or Network File System (NFS) with the binary option.

When the application code runs on the S/390 server, the entire operating system interface is handled by the Domino server code. Therefore you do not need to change the application to run on S/390.

## 1.2 Porting a C Application

Lotus also provides the ability for you to write C and C++ applications that interface to Notes databases. These applications use two different interfaces, as shown in Figure 2 on page 2.

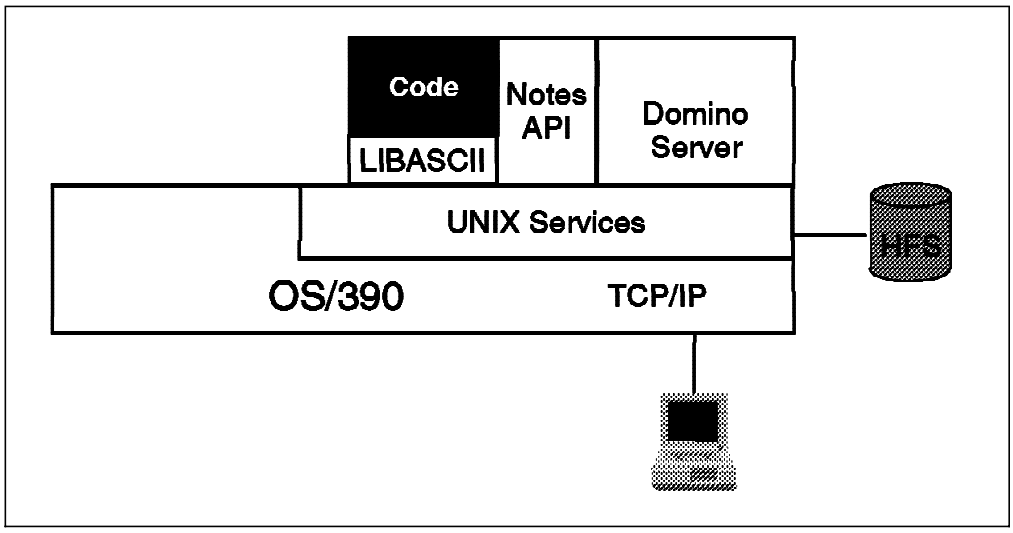



*Figure 2. Porting a C Application That Accesses a Notes Database*

- The application uses the Notes application programming interface (API) toolkit supplied by Lotus to interface to the Domino server. This toolkit has been ported to S/390 and is available from the Web. Note that there are differences in the API between UNIX and Intel server platforms, and also some minor differences between specific platforms. Therefore you may need to change your code to allow for those.
- The application also interfaces directly to the server operating system, which is OS/390 on a S/390 server. OS/390 is branded XPG4 UNIX Profile (also known as UNIX 95) compliant. Therefore, if your code uses only those interfaces, then the port should be very easy. However, it is likely that your code will use other interfaces and therefore you will need to do some code changes.

  One big difference between S/390 and other servers is that while most servers use ASCII data encoding, S/390 (and AS/400) use EBCDIC data encoding. To make it easier to port ASCII-based C/C++ applications (such as Lotus Notes applications) to OS/390, IBM provides the libascii toolkit, which handles many of the ASCII-to-EBCDIC differences for you.

## 1.3 Conclusion

A C or C++ application that uses the Notes API Toolkits must be ported to OS/390 UNIX Services. At a minimum this will require the code to be compiled with the C or C++ compiler on OS/390. It is likely that you will need to make some application changes also. The number of changes depends on how platform-independent your code is to start with.

We successfully ported Applications Partners WIT code to the Domino server for S/390. The project was completed in the timescale we predicted (ten days) with no major issues. The port was carried out by one experienced programmer from the application vendor, with environmental support from IBM. Application Partners accurately predicted most of the changes that would be needed.

OS/390 provides a true UNIX “look and feel” from an application programming viewpoint. Porting applications to OS/390 UNIX Services is not significantly different from porting applications to any UNIX platform.

# Chapter 2. Porting Project Overview

This redbook discusses a project to port the Application Partners WIT application to Domino for S/390. This chapter provides background information about the WIT product and the porting project. This will enable you to compare this porting project with any that you might be considering.

## 2.1 Description of the WIT Product

Application Partners, Inc. is a consulting and development company that services the Fortune 2000. Located in central New Jersey, and incorporated in August 1992, the company specializes in developing solutions and tools for the Internet, Lotus Notes, relational databases, and other client-server environments. Application Partners is a Sun-Java developer, a Lotus Notes Premium Partner, and an Oracle ISV partner, as well as an IBM Partners In Development member. IBM is also a customer of Application Partners.

Application Partners offers several software products which help companies to build sophisticated applications using Lotus Notes, Oracle and Web-related technologies. It has created a unique toolset for the Lotus Notes environment. Applications Partners' tools and knowledge have been very well received by the Lotus Notes community. The company has evolved from being a simple tool provider to a company that also augments the development talents of corporations across the United States. Application Partners, Inc. has more than 500 customers worldwide.

You can find details of Application Partners and its products on the Web at `http://www.application-partners.com/`.

### 2.1.1 The WIT Product

While the objective of Application Partners had been to develop a product that capitalizes on features specific to the 390 platform, such as DB2 database connectivity, the core of that product will evolve from the product known as *WIT*. In order to meet the objectives of the project set by IBM and Application Partners, they ported the current WIT product to the S/390 platform during this project.

For those companies using Lotus Notes, WIT allows them to save development time, reduce the rising costs of development and incremental training, and improve the management, deployment, and maintenance of world-class applications more effectively. WIT currently has over 44 individual functions which can be uniquely combined to meet virtually any application requirement. By simply clicking on buttons, companies can create robust applications in a fraction of the time it would normally take them to do so using standard Notes tools.

WIT is a library of Lotus Notes API-based actions that are accessible within the standard Notes environment. These actions are pre-programmed processes which are performed in a specified order, either at timed intervals or by being triggered by the occurrence of a specified event. By simply filling in forms and clicking on buttons to combine these actions, anyone can create Lotus Notes applications.

WIT's actions include:

- **Create Report in Target Database**: Collect data from one or more source documents residing within one or more source databases, and then place the collected data into one target document.
- **Create Documents in Target Databases**: Create new documents in one or more databases.
- **Delete Documents in Target Databases**: Delete documents from one or more target databases or delete the trigger document.
- **Copy Trigger Document to Target Databases**: Copy a single trigger document to one or more target databases.
- **Copy Source Documents to Target Databases**: Copy one or more source documents to one or more target databases.
- **Copy Trigger Data to Target Documents**: Collect data from a single trigger document, and copy the collected data to one or more target documents residing within one or more target databases.
- **Copy Source Data to Target Documents**: Collect data from one or more documents residing within one or more databases, and copy the collected data to one or more target documents residing within one or more target databases.
- **Find Target Documents then Copy Source Data**: Collect data from one or more source documents, identify one or more target documents residing within one or more target databases, and then copy the collected source of data to the target documents.
- **Copy Source Data to Trigger Document**: Collect data from one or more documents residing within one or more databases, and then copy the collected data to the trigger document.
- **Move Trigger Document to Target Database**: Move the trigger document to a single target database.
- **Move Source Document to Target Database**: Move one or more source documents to a single target database.
- **Move Trigger Data to Target Document**: Collect data from the trigger document and move that data to a single target document.
- **Move Source Data to Target Document**: Collect data from various source documents residing within one or more databases, and copy the collected data to a single target document.
- **Find Target Document then Move Source Data**: Identify a single target document, collect data from one or more documents, and then move the collected data to the identified target document.
- **Move Source Data to Trigger Document**: Collect data from various documents residing within one or more databases, and then move the collected data to the trigger document.
- **Modify Document**: Identify documents residing within one or more databases, and then modify the fields of the identified documents.
- **Number Generator**: Identify documents that require a numeric field to be updated using a sequentially generated number.
- **Import**

- **Export**
- **Download**: Add, Modify, Replace, Delete, Ignore and Conditional

More than 15 other actions and functions are also supplied that you can use to create Lotus Notes applications quickly and easily.

### 2.1.2 WIT Code Details

The WIT code is written in C and used the Notes C application program interface (API) toolkit. It consists of:

- 29 C programs (.c files)
- 11 header files (.h files)
- A total of 23,850 lines of code (including comments)

The code was written for the OS/2 platform and later ported to Windows NT. On those platforms it was compiled with C++ compilers.

Although the vendor isolated various operating system-dependant modules of the code to facilitate porting between environments, WIT had *not* previously been ported to UNIX platforms. Therefore some of the changes that we had to make were because of differences between the Intel and UNIX platforms. If we had started from a UNIX version of the code, then some of the items we changed would have already been in the code. However, for our exercise, it was useful to discover the issues in moving from the Intel platforms, since we believe that many of the ports will start from there.

## 2.2 The Porting Project

The objective of this porting project was to work with a software vendor to port a Notes C application to Domino for S/390. Application Partners, Inc. has the WIT product that is popular with Notes users on other platforms, especially OS/2 and Windows NT, and it was interested in doing an early port of this code.

IBM's specific objectives were:

- To understand the items that need to be considered in porting a Notes C application to Domino for S/390
- To document those considerations in this redbook to assist customers and other software vendors who wish to do similar ports
- To exercise the ported Notes C API code prior to general availability

Application Partners' specific objectives were:

- To understand the changes necessary to enable the code to run on the S/390 platform
- To complete as much of the port as possible

### 2.2.1 People Involved

The participants in this project were:

- John LaFata from Application Partners, Inc. John knew the WIT code very well and he made the code changes and tested them. John had experience with UNIX from other projects.
- Alain Atge from IBM France. Alain provided S/390 support to John and documented the stages of the project in this redbook. Alain had a good knowledge of programming techniques and C.
- Brian Macfaden from the ITSO in Poughkeepsie. Brian managed the project, provided the S/390 infrastructure and edited the final redbook.

For more information on the authors, see “The Team That Wrote This Redbook” on page ix.

### 2.2.2 The Environment

We worked with the following software:

- OS/390 Release 3 with the UNIX Services
- The IBM C/C++ Language Environment, compiler, and tools
- Lotus Domino Server Release 4.51 for S/390 Beta 101 code
- Lotus Notes C API Release 4.51 Toolkit for S/390 early code
- IBM libascii toolkit early code

Although we were working with Domino server beta code and the two toolkits had only recently been ported to S/390, we found only one error due to the software products. This was fixed and will not occur in the generally available code.

### 2.2.3 What We Achieved

The porting project was successful. Within the ten days of the project, John had modified the code so that it:

- Compiled on OS/390
- Opened a Notes database
- Opened a view in the database
- Opened documents in the view
- Read fields from a document
- Searched a rich text field to find a doclink
- Used that doclink to open the associated document
- Read fields from that document
- Created documents
- Added fields to documents
- Modified existing fields
- Used the Notes macro formula language to set values in fields
- Appended to rich text fields
- Set fonts and colors in rich text fields

John built the code changes so that the same source can be compiled on S/390 as well as on the original OS/2 and Windows NT platforms. He also retested the changed code on the original platforms to make sure that it still worked correctly, and to make sure that the code was truly portable.

John completed the port, apart from two minor errors that we did not resolve during the project:

- We could not stop the addin task by using the `quit` command on the server command line.

  Application Partners will use an alternative way to terminate the product until the problem is isolated and fixed. WIT can be stopped by putting a document in the database in a known location, or at a given time, or after it runs all the processes it is to run.

- We sometimes got memory segment violation error messages when we closed down the Domino server after running the WIT product. This problem is documented in 6.8, “Segmentation Violation When Close Domino Server” on page 48.

  WIT does not experience the same problem on other platforms and it did not occur consistently. Application Partners plans to address this area when the Notes C API code for S/390 is out of the beta program.

Since the project completed, Application Partners continues development in its own facilities.

# Chapter 3. Planning the Port

This chapter describes the OS/390 UNIX development environment that you will use to port your Notes C applications to OS/390. In it, we tell you the way in which we worked and describe other alternatives you could use. It also discusses some of the things you should think about in preparing your code for the port, the documentation that we used, and the assistance that is available to software vendors from IBM.

## 3.1 The Development Environment on OS/390

OS/390 is branded XPG4 UNIX Profile (also known as UNIX 95) compliant. OS/390 provides a true UNIX environment for applications and for application developers. Remember that this is *not a UNIX emulation*. Rather applications on the OS/390 server can call for system functions (such as getting memory or starting new tasks or processes) using two different sets of calls:

- UNIX calls defined by the UNIX standards
- Traditional OS/390 (MVS) calls

In both cases the calls are handled by OS/390, the UNIX calls being handled by the component named OS/390 UNIX Services (previously called OpenEdition). In fact an application can use both sets of calls at the same time.

OS/390 also provides a standard UNIX hierarchical file system (HFS) for storing your files. Anyone familiar with UNIX will find this very easy to work with. Even people not too familiar with UNIX will find that it is similar to the PC file system that they do know.

Note that you will need access to some OS/390 skills to tailor the server platform for the use of the UNIX programmers.

### 3.1.1 Toolkits

As already discussed, we used two toolkits to port the WIT code to Domino for S/390:

- The Lotus Notes C API Toolkit
- The libascii package for handling ASCII/EBCDIC differences

See Chapter 4, “Notes C API Toolkit and libascii Installation” on page 17 for more information on these two packages.

### 3.1.2 User Interface

OS/390 provides several different user interfaces into the UNIX environment. You can use any or all of them, depending on your preferences.

**Note:** To access the OS/390 UNIX environment, you must have a valid OS/390 user ID and password. The user ID must have a valid OMVS segment defined. The OS/390 security administrator will need to set up this ID.

#### 3.1.2.1 rlogin and Telnet

If you are familiar with UNIX, you will probably prefer the rlogin (from a UNIX workstation) and Telnet (from a PC) interfaces. These give you a standard UNIX shell, the Korn shell, licensed from Mortice Kern Systems. With these interfaces, you can invoke programs, shell scripts, and REXX execs, and have access to UNIX commands and utilities.

The terminal operates in byte (raw) mode. Data is transmitted as it is entered. It supports the vi editor and curses.

#### 3.1.2.2 OMVS

If you are familiar with TSO on OS/390, you may prefer to use the OMVS environment. You access this by typing the TSO command `OMVS`. Thus you can get into the UNIX environment from any TSO screen.

This again gives you the Korn shell, but there are some differences from the rlogin and Telnet environments including:

- The screen operates in line mode.
- You can scroll the screen backward and forward. This is useful, for example to review the output messages from a compile.
- You can retrieve previous commands to edit and reissue them.
- The vi editor is not supported. However, there is an editor called oedit that you can use to edit files in the UNIX file system.
- This environment does not support curses.

Thus, while many operations can be performed in OMVS, you may find that some things do not work because it is more limited.

#### 3.1.2.3 Interactive Shell (ISHELL)

OS/390 also provides a full screen ISPF dialog interface into the UNIX environment. This is called the interactive shell and it is accessed by starting the ISPF application `ISHELL`. It allows you to work with files and directories using an ISPF interface. If you know ISPF and do not know UNIX well, you may find this interface useful as it allows you to do many tasks without knowing the UNIX commands.

### 3.1.3 Transfer Files from Workstation

OS/390 supports different ways to transfer files between your workstation and the S/390 server:

**ftp** You can use the file transfer program (ftp) to transfer files between your workstation and the S/390 server. You can transfer them directly into the HFS, or into standard OS/390 sequential and partitioned data sets. To transfer them directly into the HF, you need the OpenEdition Application Feature installed and active on the S/390 server. See *OS/390 Release 3 OpenEdition Installation and Customization*, SG24-2087. This is planned for first quarter 1998.

**NFS** The Network File System (NFS) allows you to mount a directory in the OS/390 UNIX HFS as a UNIX directory or PC disk. You then edit the files as though they are on your workstation, but in fact they are on the S/390 server. NFS

requires setup on OS/390 (see *OS/390 Release 3 OpenEdition Installation and Customization*, SG24-2087 and on the workstation (see your workstation documentation).

**Send/receive** You can use any other file transfer product to transfer files between your workstation and OS/390. Often these put files into OS/390 sequential and partitioned data sets. You then use the TSO OPUT and OGET commands, or the ISHELL, to copy these files to and from the UNIX HFS.

To transfer the source code to OS/390, we used ftp and uploaded the files directly into the UNIX Hierarchical File System (HFS) on OS/390. We uploaded them in ASCII (text) mode.

**Note:** You may wish to put your application code in a separate UNIX directory. Consider also whether you wish this directory to be in a separate OS/390 HFS data set. If so, that data set will need to be defined and mounted by the OS/390 support staff.

### 3.1.4 Editing Files

You can edit files:

- Using the vi editor from rlogin or Telnet.
- Using oedit, which is supplied as part of OS/390 UNIX Services. The oedit editor allows you to edit files in the Hierarchical file system (HFS). It uses the ISPF/PDF Edit facility.
- On your workstation using any editor you prefer. You then transfer the files to OS/390 using one of the methods described earlier.

John brought the WIT code on diskette. He did all of the code editing on an OS/2 workstation using an OS/2 text editor. This is the environment that John was most familiar with. It also meant that he was editing the source in ASCII, and he did not need to concern himself with ASCII/EBCDIC differences when editing.

When we edited files on OS/390, we used the oedit editor. We used this to edit the `.profile` file and the make file.

### 3.1.5 Compiling

You must compile your code on the OS/390 platform using one of the standard OS/390 C compilers. These produce the binary code necessary to run on OS/390. Note that it is the same C compiler that you would use for any other C programs on S/390. There is no difference in the way a *UNIX* C program is compiled from a *non-UNIX* C program. The only difference between them is the calls that they use to get services from the operating system.

You have the following facilities:

- A choice of the C89, Cxx and C++ compilers. C89 carries out the ANSI C 89 standard and is more rigorous than Cxx.
- The make command and makefiles.
- The dbx debugger.
- Environment variables.
- Archive files produced by the UNIX commands: `ar, pax and tar.`

We drove all of the compiles from the OS/2 workstation. We logged on to TSO, used the OMVS command to go into the UNIX environment, and issued the compiler and other UNIX commands from there. Alternatively we could have used a Telnet session from our PC, or rlogin from a UNIX workstation. We found OMVS a better interface for the compiles as it allows you to scroll backward and forward through the error messages.

To compile and link the code, we used the os390.mak file that we used to run the intro sample code. See 4.3.7, “Building and Running the intro Sample” on page 24. We modified the file names to reflect the WIT code.

We used the UNIX make command to run the compiles. Figure 3 shows a sample make command:

- os390.mak is the name of the make file.
- 2>&1 | tee test routes stderr output to both the console and the file test. With this method, the error messages are presented in context with stdout which sometimes aids debugging.

```
make -f os390.mak 2>&1 | tee test
```

*Figure 3. Sample Make Command*

We used the C89 OS/390 Compiler. This is a C compiler that keeps fairly strictly to the ANSI C89 standard. As a result we got a number of compiler errors that John did not get when he compiled the code on OS/2 or NT. The compilers that he used on those platforms did not enforce the ANSI C89 standard so rigorously. We could have chosen to use a less rigorous compiler, such as the CC compiler, or we could have tried the C++ compiler. However we chose to continue with the C89 compiler and we therefore fixed the compiler errors that C89 gave us.

### 3.1.6 Testing the Code

We tested the code from a Telnet session. We could have used rlogin from a UNIX workstation. We found that OMVS did not work for testing of the WIT code. There are some differences in the OMVS environment - see 3.1.2.2, “OMVS” on page 10.

Therefore you may or may not be able to test using OMVS. We did not spend time trying to work out what the issue was with WIT. We just used Telnet and the testing ran fine.

### 3.1.7 Regression Testing

John's objective was to finish with a single set of source code that would run on OS/390, OS/2 and Windows NT. He brought a laptop computer with him that had OS/2 Warp Version 3 and the IBM VisualAge C++ Compiler Version 3. He used this as a reference system. Periodically he took the changed code and compiled and tested it on OS/2 to make sure that it still ran there.

## 3.2 Modifications to Your Code

It is likely that you will need to make some modifications to your code. In later chapters we will describe the changes needed to Application Partners' WIT code. As you read the rest of the book, remember that the changes you need to make depend on what the state of your code is. Possible changes are:

- Changes because S/390 uses EBCDIC encoding and most other platforms use ASCII encoding. You will almost certainly need to do at least some changes to allow for this. See 3.2.1, “ASCII/EBCDIC Differences.”
- Changes because of compiler differences. The C89 compiler that we used on OS/390 is enforced to the C89 ANSI standard more rigorously than the C++ compilers used to compile the code on OS/2 and Windows NT. Therefore some lines of code that worked on the Intel platforms were rejected by the C89 compiler and needed to be changed.
- Changes because of platform differences. OS/390 uses different header files than other platforms (see 5.2.3, “Header File Differences” on page 28). A few functions are not available. It does not have a resource compiler.
- Changes because of differences between UNIX and Intel platforms. Notes API structure lengths are different on UNIX and on Intel. Host format rich text is not supported on UNIX platforms. File names are case-sensitive on UNIX, and path names use a slash rather than a backward slash.
- Changes because of the products that your application uses. Where your application interfaces with other products, you need to check whether those interfaces are available on OS/390. If not, you may need to change the interfaces that you use.

Your code may be coded so that these differences do not affect it, or it may be coded very specifically to one server type, in which case you will have more work to do.

As you approach the port, you can follow one of two paths:

- You could run a code checker to find out what the likely changes are.
- You could use the preceding list, and the rest of this book, to assess the likely changes based on what you already know of the code.

John took the second approach and, from his knowledge of the code, he predicted almost all of the likely areas for modification before we started. He then compiled the code on OS/390 and started from there.

### 3.2.1 ASCII/EBCDIC Differences

From our experience of porting other applications to OS/390, we know what the differences between ASCII and EBCDIC are, and the likely changes you will need to make to your code.

You will almost certainly need to change the way that you handle strings. Your strings and code will compile on other platforms in ASCII. When you port to S/390, they will by default compile in EBCDIC, and you will end up with incorrect string handling.

To help you with this, IBM has produced a package called libascii. The libascii package provides an ASCII interface layer for some of the more commonly used C/C++ run-time library functions. It supports ASCII input and output characters

by performing the necessary iconv() translations before and after invoking the C/C++ run-time library functions. The OS/390 V1R3.0 C/C++ compiler predefined macro __STRING_CODE_SET__="ISO8859-1" generates ASCII characters rather than the default EBCDIC characters. Using this with the libascii code provides an ASCII-like environment.

As a minimum, you will wish to compile your code using libascii to handle many of the differences. This involves:

- Adding a macro to the make file to compile the strings in ASCII
- Include the libascii header file in your source code
- Link edit your code with the libascii archive file

For more information about our use of libascii, see 5.1, “Modifying the Code” on page 27. For more complete information about libascii, see Appendix B, “OS/390 OpenEdition libascii Read File” on page 57.

#### 3.2.1.1 Other ASCII/EBCDIC Differences

For the WIT code, we needed only to link it to the libascii package and all of the ASCII/EBCDIC differences were taken care of. However, there are other differences that you need to be aware of. If your application uses these, then you need to make further modifications. The differences are:

***Hardcoded ASCII Characters:*** Avoid using hardcoded values or depending on the values of characters in C code and shell scripts. For example, a program might use \012(octal) instead of \n. It might hardcode special characters such as carriage return line feed (crlf), new line (nl) or tab (tab).

***Using the High-order Bit of a Character for Some Special Purpose:*** You can use the high-order bit of a character for a special purpose (such as a flag) in ASCII because only seven bits are necessary for all the printable characters. EBCDIC uses all eight bits, so the results will be different.

***Assuming the Alphabet Is Contiguous:*** The alphabet (a-z) is encoded as contiguous hexadecimal values in ASCII, so you can add one to “i” and get “j.” This is not true in EBCDIC, where there are three noncontiguous groups of letters.

***Using Code Generated by lexx or yacc:*** Packages may contain C code that was generated by the lexx or yacc utilities. This code will probably contain ASCII dependencies and will not work on OS/390. It needs to be generated on OS/390 by re-running the utilities.

***Applications That Talk to Arbitrary Remote Systems Using Sockets:*** This would include an ftp client. These applications typically have to assume that all text they receive is ASCII and they send out all text as ASCII. They have to convert the data locally as they go along.

***Code That Relies on Byte Order of Data:*** Code that relies on byte order of data may not be portable. PC systems are “little endian” (that is, the leftmost byte is the most significant). S/390 and most UNIX systems are “big endian.” This typically affects integer and floating point data. If an application is responsible for transferring such data between platforms, you need to either (1) write data exchange logic or (2) translate to text, transfer as text, and then recreate as binary.

***Floating Point:*** There are differences between IEEE floating point format and S/390 floating point format. See Appendix B, “OS/390 OpenEdition libascii Read File” on page 57.

For more information on these differences, see the Web site: `http://www.s390.ibm.com/oe/bpxa1p03.htm`.

## 3.3 Documentation

We used the following documentation in our port:

- The Notes C API Toolkit readme file, which is shipped in the tar file. For your information Appendix A, “API 4.5.1 Code Readme File” on page 51 contains the text of the readme file at the time of publication of this book. You should read the file shipped with the toolkit for the latest information.
- *Notes C API 4.51 User Guide (api451ug.nsf)*, which is shipped in the Notes C API Toolkit tar file. This is a guide to the use of the Notes C API. If you are porting a C application to S/390, you should particularly review:
  - Chapter 3 “Platform Specifics” section “Building UNIX Applications.” You should review the entire section. A particularly relevant part is called “Porting Notes C API Programs to UNIX from OS/2 and Windows.”
  - Chapter 12 “Platform Issues,” which discusses Notes canonical format and platform-specific naming conventions
- *Notes C API 4.51 Reference (api451re.nsf)*, which is also shipped in the Notes C API Toolkit tar file. This is reference documentation for the Notes C API. We did not need to refer to this during our port.
- The libascii package readme file, which is shipped in the tar file. For your information Appendix B, “OS/390 OpenEdition libascii Read File” on page 57 contains the text of the readme file at the time of publication of this book. You should read the file shipped with the package for the latest information.
- *Porting Applications to the OpenEdition MVS Platform, GG24-4473*, which describes the port of a different vendor application to MVS, the predecessor of OS/390. Some of the information in this book is now out of date. However it still includes much useful advice on running a porting project and some other platform differences you may need to consider.

## 3.4 Assistance from IBM

IBM is able to assist solution developers in developing or porting applications to the S/390 platform through the S/390 Partners in Development program. We have Solution Partnership Centers around the globe which provide worldwide remote support, including:

- Hursley, England
- Boeblingen, Germany
- San Mateo, California
- Waltham, Massachusetts
- Dallas, Texas

These centers have already assisted software vendors to port their applications to Domino on S/390.

For more information about S/390 Partners in Development see the Web sites:

- http://www.s390.ibm.com/products/s390da/only.html for S/390 Partners In Development program information
- http://www.s390.ibm.com/products/s390da/lotus.html for information on the Lotus Domino initiative

# Chapter 4. Notes C API Toolkit and libascii Installation

This chapter describes how we installed and tested the Notes C API Toolkit and the libascii code. We installed the libascii code first. As already discussed, the libascii code is useful to many applications that are ported to S/390 using the OS/390 UNIX services.

## 4.1 The Directory Structure

When we installed the two packages, we used the directory names that are assumed in the readme files for the two packages. This is shown in Figure 4.

You will already have the /usr directory on your system. libascii is installed off that directory. /usr/lpp/lotus is the directory where Lotus Domino is installed, and the Notes C API Toolkit is installed off that directory.

You can use different directories if you wish. However, the directory names shown are a logical path for others to follow.

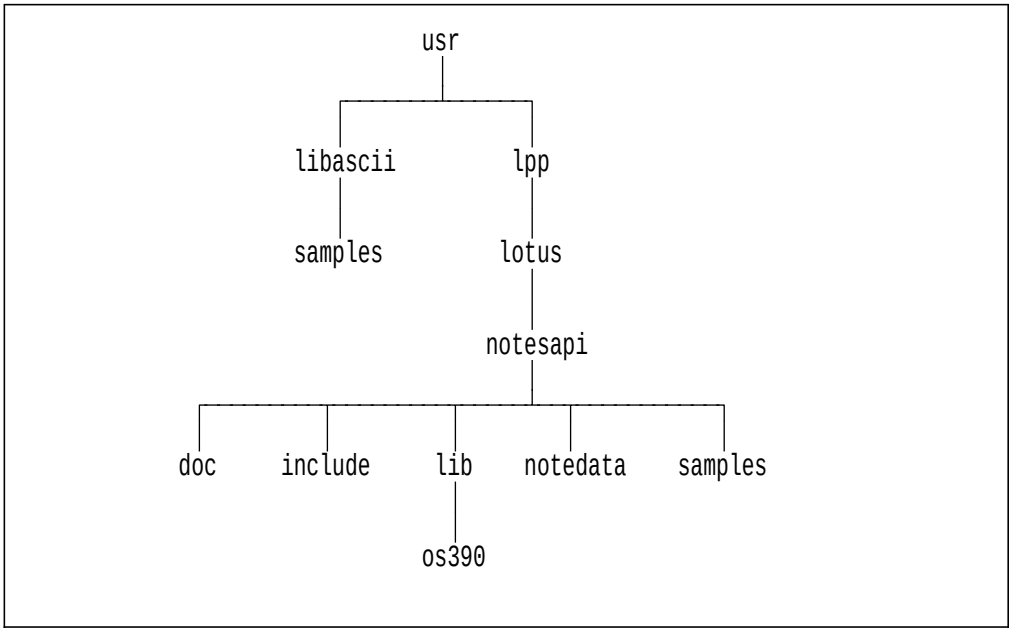



*Figure 4. Directory Structure after Installing Toolkits*

Within /usr/lpp/lotus/notesapi you will see the following files and subdirectories:

**readme.390** Read this file for the installation instructions and the latest information.

**doc** Contains the two Notes C API Toolkit documentation databases.

**include** Contains header files (*.h).

**lib/os390** Contains *.o files.

**notedata** Contains Notes databases used by the sample programs.

**samples** Contains sample programs. The contents of this directory are shown in Figure 5 on page 18.

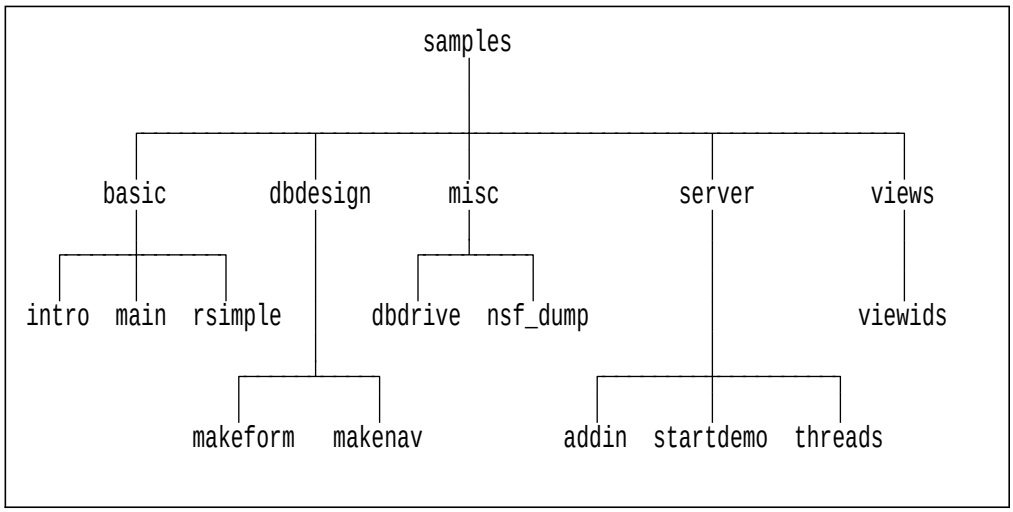



*Figure 5. Directory Structure of Notes C API Toolkit Samples*

### 4.1.1 File Space Needed

For the versions of the two packages that we installed, the file space requirements are shown in Table 1. Column two shows the size of the tar file. Column three shows the additional space required when the tar file is unpacked.

*Table 1. File Space Needed by Packages*

| Package | Tar file size | Unpacked size |
|---|---|---|
| libascii | 60K | 280K |
| Notes C API Release 4.51 | 5,489K | 17,100K |

You need to ensure that your file system has sufficient space for these files. You may wish to allocate new HFS data sets for these two products and mount them to the mount points described. See *Lotus Domino Server Release 4.5 on S/390: Installation, Customization and Administration*, SG24-2083. for information on how to allocate new HFS data sets. This manual is due to be published by the end of first quarter, 1998.

Once the tar files have been unpacked, you can delete the tar files to save space.

## 4.2 libascii Installation

The libascii package helps you port ASCII-based C applications to the EBCDIC-based OS/390 UNIX environment.

The C/C++ Run-Time Library functions support EBCDIC characters. The libascii package provides an ASCII interface layer for some of the more commonly used C/C++ Run-Time Library functions. libascii supports ASCII input and output characters by performing the necessary iconv() translations before and after invoking the C/C++ Run-Time Library functions.

The OS/390 V1R3.0 C/C++ compiler predefined macro `__STRING_CODE_SET__="ISO8859-1"` will generate ASCII characters rather than the

default EBCDIC characters. Using this with libascii will provide an ASCII-like environment.

See Appendix B, "OS/390 OpenEdition libascii Read File" on page 57 for a list of the functions provided in libascii, some limitations, and frequently asked questions.

### 4.2.1 Downloading the libascii Package

The libascii package can be downloaded from the Web at URL:

```
http://www.s390.ibm.com/products/domino
```

We downloaded it to a PC and saved it with the file name `libascii.tar`. We also saved a copy of the Web page on our PC to use as installation instructions.

### 4.2.2 Uploading the libascii Package to OS/390

Next we uploaded the tar file to the S/390 server. First we created a new directory `/usr/libascii`. Then we uploaded the tar file to the S/390 server using ftp and placed it in the directory `/usr/libascii`. The command sequence to do this is shown in Figure 6.

```
ftp wtsc54                      <==  wtsc54 is the TCP/IP name of the
                                     S/390 server
<userid>                        <==  Valid OS/390 user ID
<password>                      <==  Password

md /usr/libascii                <==  Make new directory on S/390
cd /usr/libascii                <==  Set directory on S/390
lcd d:\                         <==  Set directory on workstation (PC)

binary                          <==  Upload in binary mode

put libascii.tar                <==  Upload the file
```

*Figure 6. FTP Commands to Upload Package to S/390*

### 4.2.3 Untaring the libascii Tar File

To unpack the tar file we went to the OS/390 UNIX shell, using the TSO OMVS command, and issued the commands in Figure 7.

```
cd /usr/libascii

tar -xvfoz libascii.tar
```

*Figure 7. Untar libascii*

This resulted in the libascii source files, make file and readme file being installed in the `/usr/libascii` directory and the `samples` subdirectory. Read the `readme` file as it contains the latest information on libascii, including installation information. Appendix B, "OS/390 OpenEdition libascii Read File" on page 57 lists the readme file at the time of writing.

You could now delete the tar file to save space.

### 4.2.4 Building the libascii.a Archive File

We used the makefile file that is provided with the package to build the libascii.a archive file from the libascii source files. The command sequence is shown in Figure 8.

```
cd /usr/libascii

make
```

*Figure 8. Making the libascii.a Archive File*

The make ran successfully.

**Note:** On one of our tests we got the following error message from the make command: `make: Error -- FSUM9383 Configuration file '/etc/startup.mk' not found`

We fixed this by copying the file "startup.mk" from the `/samples` directory to the `/etc` directory. The command was `cp /samples/startup.mk /etc` directory. We then repeated the make command and it completed successfully.

### 4.2.5 Building and Running the Samples

To check the installation of libascii you should now build and test the sample programs in the `samples` subdirectory. These two programs are C programs with no access to a database. They perform a few operations on strings and displaying strings using the `printf()` function.

The command sequence to make the two files is in Figure 9. The output from the two sample programs is shown in Figure 10 and Figure 11 on page 21.

```
cd /usr/libascii/samples

make
```

*Figure 9. Build the libascii Samples*

```
# sample1
  Letter A converted to lower case is:a
  Letter \ converted to lower case is:\
  Letter Z converted to lower case is:z
#
```

*Figure 10. Output from sample1*

```
# sample2
The value of argc is: 1

Comand line arguments:

    argvÝ0": 'sample2'

Environement Variable PATH=/bin:.
:hello, world:
:hello, world:
:hello, wor:
:hello, world:
:hello, world:
:hello, world   :
:     hello, wor:
:hello, wor     :   :hello, wor     :
:hello, wor     :   :hello, wor     :
Print the letter z after :z
Print the letter z after :         z
 45 =45        :
 45.678 =45.678    :

List the files in the current directory using opendir/readdir

.                   ..                  makefile            sample1
sample1.c           sample1.o           sample2             sample2.c
sample2.o           sample3             sample3.c           sample3.o
sample4             sample4.c           sample4.o
#
```

*Figure 11. Output from sample2*

### 4.2.6 Next Steps

If you plan to use libascii for other applications, see Appendix B, “OS/390 OpenEdition libascii Read File” on page 57 and `/usr/libascii/readme` for further steps that you need to do. We will discuss them later in the book as appropriate. In particular, we describe the changes we made to the WIT application in 5.1, “Modifying the Code” on page 27.

## 4.3 Notes C API Toolkit

The Notes C API Toolkit is provided by Lotus as a tool to enable C applications to access Notes databases.

### 4.3.1 Downloading the Notes C API Package

The Lotus Notes C API Release 4.51 for OS/390 can be downloaded from the Web at URL:

`http://www.s390.ibm.com/products/domino`

You can also find it on the Lotus Web site at URL:

`http://193.164.160.162/ldw.nsf/Data/Document1289`

We downloaded the package to a PC and saved it with the file name `capi451.tar`. We also downloaded the file `readme.390` to use as installation instructions. We have included a copy of that file at the time of writing in Appendix A, “API 4.5.1 Code Readme File” on page 51.

### 4.3.2 Uploading the Notes C API Package to OS/390

We uploaded the tar file to the S/390 server using ftp and placed it in the existing directory /usr/lpp/lotus. Assuming the ftp session in Figure 6 on page 19 is still open, the command sequence to do this is shown in Figure 12.

```
binary                          <==  Upload in binary mode

cd /usr/lpp/lotus               <==  Set directory on S/390
lcd d:\                         <==  Set directory on workstation (PC)

put capi451.tar                 <==  Upload the file
```

*Figure 12. Upload Notes C API Package to S/390*

### 4.3.3 Untaring the Notes C API Tar File

To unpack the tar file, we went to the OS/390 UNIX shell and issued the commands in Figure 13.

```
cd /usr/lpp/lotus

tar -xzvof capi451.tar
```

*Figure 13. Untar Notes C API*

The C API Toolkit is unpacked into a directory structure under the subdirectory notesapi of the current directory /usr/lpp/lotus. This directory structure is shown in Figure 4 on page 17.

Read the readme file /usr/lpp/lotus/notesapi/readme.390, as it contains the latest information on the Notes C API, including installation information.

You could now delete the tar file to save space.

### 4.3.4 Setting the Environment

We now set up our environment as stated in the Notes C API Toolkit readme file. There are some environment variables that need to be set whenever you will use the Notes C API. The easiest way to do this is to put the variables into a .profile shell script in the home directory of the user ID that will be using the Notes C API Toolkit. We added the lines shown in Figure 14 to the .profile shell script in our directory. The Notes C API readme.390 file explains the reason for each line - see Appendix A, “API 4.5.1 Code Readme File” on page 51.

```
export LIBASCII_PATH=/usr/libascii
export LOTUS=/usr/lpp/lotus
export Notes_ExecDirectory=/usr/lpp/lotus/notes/latest/os390
export PATH=$PATH:$Notes_ExecDirectory
export PATH=$PATH:/notesdata
export LIBPATH=$LIBPATH:$Notes_ExecDirectory
```

*Figure 14. Setting Environment Variables for Notes C API*

Now make this environment active in one of these ways:

- Issue the UNIX command `. .profile` (a period, then a space, then a period followed by profile with no space).
- Log off the UNIX environment and log on again.

**Attention**

It is very important that you set these variables correctly. We made some errors in our early attempts. The results were unreadable characters in response to running the sample programs and also the WIT program. The messages did not point to the variable that was set incorrectly, so it was hard to find the problem. To make sure that you have set the variables correctly, follow 4.3.5, “Verifying the Environment” carefully.

### 4.3.5 Verifying the Environment

As noted, you need to verify that the environment is correct. We recommend using two methods. The first is to display the variables using the `echo` command as shown in Figure 15. Make sure that the output includes the correct libraries, as described in `/usr/lpp/lotus/notesapi/readme.390`.

```
echo $LIBASCII_PATH
echo $LOTUS
echo $Notes_ExecDirectory
echo $PATH
echo $LIBPATH
```

*Figure 15. Verify Environment with Echo Command*

The second method is to verify the location of some key files using the `whence` command as shown in Figure 16. If the file is found, the full path name will be displayed. Check that each file is found in the correct library. If the file is not found, there will be no response to the command, which means that you have made an error.

```
whence $LIBASCII_PATH/_Ascii_a.h
whence $LIBASCII_PATH/libascii.a

whence $LOTUS/notesapi/include/lapiplat.h
whence $LOTUS/notesapi/lib/os390/notes0.o
whence $Notes_ExecDirectory/libnotes.x
whence $Notes_ExecDirectory/libnotes
```

*Figure 16. Verify Environment with Whence Command*

### 4.3.6 Copying the Notes C API Toolkit Documentation

The Notes C API Toolkit documentation is provided in two Notes databases that are unpacked from the tar file into the directory `/usr/lpp/lotus/notesapi/doc`:

**api451ug.nsf** *Notes C API 4.51 User Guide*

**api451re.nsf** *Notes C API 4.51 Reference*

These databases need to be copied to a location where they can be viewed. We copied them to the doc subdirectory of the Domino server data directory (/notesdata on our system). We used the UNIX command:

```
cp /usr/lpp/lotus/notesapi/doc/* /notesdata/doc/.
```

Note the period immediately following the final slash. You can then delete the files from the original directory to save space.

Alternatively, you could download these two databases to your Notes workstation.

### 4.3.7 Building and Running the intro Sample

To check the installation of the Notes C API Toolkit, you should now build and test at least one of the sample programs in the /usr/lpp/lotus/notesapi/samples subdirectory. We built and ran the intro sample using the command sequence in Figure 17.

```
cd /usr/lpp/lotus/notesapi/notedata                <==  see note 1
cp * /notesdata/.

cd /usr/lpp/lotus/notesapi/samples/basic/intro     <==  see note 2
make -f os390.mak

intro                                              <==  see note 3

Lotus Notes C API Release 4.51 Sample Application


The title for the database intro.nsf is...

API Test Database (intro)
```

*Figure 17. Build and Test the intro Sample Program*

**Notes:**

1. The intro program uses the Notes database intro.nsf which is supplied with the C API toolkit in directory /usr/lpp/lotus/notesapi/notedata. This database must be copied to the Domino server data directory (/notesdata on our system) for the intro program to run successfully. The Domino server does not need to be started. These commands copy all eight databases from the input directory.
2. We make the intro program.
3. We run the intro program. The expected output is shown.

**Note:** During our testing, we initially got a single line of unreadable characters when we ran intro. This was caused by incorrect path statements. See 4.3.5, “Verifying the Environment” on page 23 for information on checking that your path statements are correct. Also check that you can find the intro.nsf database with the command:

```
whence /notesdata/intro.nsf
```

### 4.3.8 Other Sample Programs

The other samples are built similarly, though we did not try them. Each sample directory contains a `readme.txt` for that sample. The Notes C API 4.51 User Guide contains specific instructions for each sample application provided.

**Note:** OS/390 supports only those samples that execute in the server environment, since we do not support the Notes client on OS/390.

# Chapter 5. Compiling the Code

This chapter describes our experiences in compiling and linking Application Partners' WIT code on the S/390 platform. We have included the problems we encountered and the solutions we used to solve them.

## 5.1 Modifying the Code

Before compiling the WIT code, we modified it to link in to the libascii package. As described in the package readme file (see Appendix B, "OS/390 OpenEdition libascii Read File" on page 57), we did the following:

- We identified which source programs use string functions and added the statement: `#include "_Ascii_a.h"` to those source files. Note that this must be the *last* #include statement in the file.
- We added the compiler option: `-D__STRING_CODE_SET__="ISO8859-1"` to the make file to cause the compiler to generate all strings defined in the programs in ASCII rather than EBCDIC format.
- We linked libascii to the WIT code by creating a make file based on a combination of the makefiles provided with:
  - The libascii `samples` program
  - The Notes C API `intro` program

  A portion of the makefile is shown in Figure 18.

```
# INCDIR specifies where to search for include files.  If compiling
# in ascii format, then the LIBASCII headers are required.
.IF $(CODE_SET) == ascii
INCDIR = -I$(LOTUS)/notesapi/include -I$(LIBASCII_PATH)
.ELSE
INCDIR = -I$(LOTUS)/notesapi/include
.END
   ...
   ...
# LIBRARY specifies the location of external libraries needed for
# object resolution.
.IF $(CODE_SET) == ascii
LIBRARY = $(LIBASCII_PATH)/libascii.a $(NOTESDIR)/libnotes.x
.ELSE
LIBRARY = $(NOTESDIR)/libnotes.x
.END
```

*Figure 18. The makefile on OS/390*

  The linking of the libascii product is performed by linking the libascii.a module to WIT during the link portion of the compilation.

We then compiled the code.

## 5.2 C89 Compilers Differences

The OS/390 C89 compiler meets the ANSI C89 standard. It is therefore different from some other C compilers. There were differences between the C89 compiler and the IBM VisualAge C++ Compiler Version 3 that was used to compile the code on OS/2. The ones that we found are described here. It was fairly easy to change the code so that it compiled without errors.

### 5.2.1 Coding Rules for Comments

With the C89 compiler default options, the C++ double forward slash (//) comments delimiter is not supported. However, the C89 compiler will recognize // as comments if you specify the `-Wc, SS` compiler option.

We were not aware of this option, so we used a REXX exec to change all of the // comments to `/* ....*/`. Note that by making the change manually, we had to change the comments in all the `.c` files and all the `.h` (header) files. We would use the compiler option next time to avoid making these changes.

### 5.2.2 Slash in Path Names

The code contained some include statements with file path names such as:

```
#include "<sys\stat.h>"
```

The backward slash is correct on an Intel platform, but on a UNIX platform a forward slash must be used:

```
#include "<sys/stat.h>"
```

### 5.2.3 Header File Differences

There are some differences between the C89 compiler and many other compilers in the use of header files. Applications that are not POSIX- or XPG4-compliant may include headers that are not supported by OS/390 UNIX application services. Porting an application that does not conform to those standards requires that you inspect any headers that may not be present on OS/390 and determine whether the application really requires them. As you know, headers can contain all kinds of things, from macros that simply exist for convenience to prototypes for functions that may or may not exist on a particular UNIX system.

Here is a list of some headers that you will not find on OS/390 (this list is not comprehensive). It was extracted from the URL:

`http://www9.s390.ibm.com/oe/bpxa1hdr.html`

**access.h** OpenEdition's equivalent interfaces are in *unistd.h* as defined in POSIX and XPG4.

**ar.h** No equivalent at this time.

**arpa/ftp.h** No equivalent; you can “borrow” a file from a UNIX system and use it.

**cur01.h** This header is not standardized; replace it with *curses.h*.

**dir.h** OpenEdition supports *dirent.h* as defined in POSIX and XPG4.

**macros.h** No equivalent at this time.

**select.h** Use *sys/time.h* as defined in XPG4 V2. This header contains the prototype for select() and macros such as FD_SET, among other things.

**sys/ldr.h** No equivalent at this time.

**sys/mntctl.h** No equivalent at this time.

**sys/mode.h** This header is non-portable. We use *modes.h*, but the standards do not specifically refer to this header. An include for *fcntl.h* is more portable.

**sys/param.h** This header is often unnecessary. Try removing the includes for it and see what falls out.

**sys/ptrace.h** Although we do not have this header file, we have a kernel interface (BPX1PTR; see the Callable Services book). The main reason we have this callable service is for the dbx debugging, which was ported from AIX. Much of what AIX's *sys/ptrace.h* defines shows up in the assembler macro BPXYPTRC. It should be possible to create a header file based on the macro.

**sys/reg.h** No equivalent at this time.

**sys/vmount.h** No equivalent at this time.

**sys/vnode.h** No equivalent at this time.

**termio.h** OpenEdition supports *termios.h* as defined in POSIX and XPG4.

**usersec.h** No equivalent at this time.

**userpw.h** No equivalent at this time.

The OS/390 header files are listed in *OS/390 C/C++ Run-Time Library Reference*, SC28-1663.

In our port, we added platform-specific include statements to the code to handle header file differences. This is shown in Figure 19.

```
#ifndef OS390
   #include   <malloc.h>       /* requested function calls now defined
                                  in stdlib.h */
   #include   <io.h>           /* we did not need it on OS390   */
   #include   <dos.h>
   #include   <share.h>
#endif
#ifndef OS390
    #include  <process.h>      /* replace with spawn.h    */
#else
    #include  <spawn.h>
#endif
```

*Figure 19. Precompiler Directives for Header Files*

You can use the `grep` UNIX command to search the code files for a text string; for example: `grep "string" *.h`.

We also needed to change some of the header files to redefine some functions. For example, the MAX_PATH definition is not in the ANSI C89 standard and is not known to the C89 compiler. The equivalent function is _POSIX_PATH_MAX.

So we had to add the following statement to all the source files that used the MAX_PATH definition:

```
#define MAX_PATH  _POSIX_PATH_MAX
```

For more information, see *Porting Applications to the OpenEdition MVS Platform, GG24-4473* section 5.5 *Header Files*.

## 5.2.4 Casting Changes

There are differences in casting behavior between the C89 compiler and other C compilers. The C89 compiler seems to be more strict than other C compilers. Following is the list of the casting problems we encountered and the way we solved them. This is not a complete list of all the casting differences that may exist.

**Note:** After making all these changes, we recompiled the code on OS/2. It ran error-free. Therefore, by using the correct form of casting, your code can compile and run on all platforms.

### 5.2.4.1 Increment and Cast

Message ERROR CBC3025 ./doc.c:1321 Operand must be a modifiable lvalue.

Code ((FILEOBJECT*)value_ptr)++;

Cause We encountered a problem in the order that the compiler applied the increment and cast operations. In this case, it looks like an intermediate variable is created and used by the compiler. We apparently cannot perform assignment (write) operations on such a variable.

Solution In this example, the increment operation is done before the casting:
(FILEOBJECT*)value_ptr++;

### 5.2.4.2 Increment and Assign to a Casted Pointer

Message ERROR CBC3025 ./findkey.c:228 Operand must be a modifiable lvalue.

Code *((WORD *)item_value_ptr)++ = TYPE_NUMBER;

Cause The problem occurs when we try to increment and assign a casted pointer. As in the previous case, the compiler generates and uses an intermediate variable, and it is not authorized to perform an assignment operation on an intermediate variable.

Solution Separate these actions to perform assignment on the user variable:
item_value_ptr = (WORD *)item_value_ptr +1;
*(WORD *)item_value_ptr = TYPE_NUMBER;

### 5.2.4.3 Casting and Assigning

Message ERROR CBC3025 ./witfuncs.c:499 Operand must be a modifiable lvalue.

Code old_ptr = (char *)value_ptr = pformula_str;

Cause As in the previous case, we are trying to perform an assignment operation on an intermediate variable.

Solution Separate these actions to perform assignment only on the user variables:
value_ptr = (char *)pformula_str;

```
old_ptr =  (char *)value_ptr;
```

#### 5.2.4.4 Cannot Assign a Void Pointer to an Integer Array

Message ERROR CBC3068 ./workflow.c:91 Operation between types "int(*)(void*)" and "int(*)(struct {...}*)" is not allowed.

Code `action_function[0] =  cmd_trd_func;`

Cause `action_function` is an integer array that is used to hold pointers to functions. The array is declared as `int(*array[N])(void*)`. `cmd_trd_func` is the name of the function whose pointer we want to store in the array. On OS/390, the function pointer is implemented as a pointer to a structure of some type, which is why we see `struct{...}*` in the error message. We can assign the pointer to the structure to the integer array by casting it to a generic pointer `void*`.

Solution Cast the pointer using `(int(*)(void*))` before assigning it:
`action_function[0] =  (int(*)(void*))cmd_trd_func;`

#### 5.2.4.5 Redundant Casting

Message ERROR CBC3275 ./doc.c:374 Unexpected text USHORT encountered.

Code `size =  (INT) (USHORT) (*size_ptr);`

Cause This line is casting twice with both (INT) and (USHORT). It therefore has redundant casting. (USHORT is an unsigned short integer).

Solution Remove the (USHORT) casting and it gives the same result:
`size =  (INT) (*size_ptr);`

To be sure to get the same sign behavior, another solution is:
`size =  (INT) ((USHORT) (*size_ptr));`

### 5.2.5 Mismatch in Operand Types

Message WARNING CBC3068 ./readinfo.c:591 Operation between types "unsigned char*" and "char*" is not allowed.

Code `unsigned char * data_ptr;`

Cause The compiler treats all char as unsigned char by default.

Solution Remove the redundant word `unsigned`:
`char * data_ptr;`

### 5.2.6 strupr Function Is Not Available

Message ERROR CBC3280 ./misc.c:1641 Function argument assignment between types const char*" and "int" is not allowed.

Code `strcpy(newobjectName, _strupr(objName));`

Cause `strupr(char *)` is not found. strupr is not defined by either POSIX/XPG4 or ANSI, and the C89 compiler does not support it.

Solution We developed our own equivalent function based on toupper():

```
#if !defined(OS390)
      strcpy(newobjectName, _strupr(objName));
#else
      strcpy(newobjectName, strupr(objName));
#endif
```

```
      .
      .
#ifdef OS390
        char * strupr(char * string){
           int i;
           for (i=0;i<strlen(string) ;i++ ) {
              string[i]=toupper(string[i]);
           } /* endfor */
           return(string);
        }
#endif
```

### 5.2.7 Identifiers INT, FLOAT and CHAR Not Declared

Message ERROR CBC3045 ./misc.c:159 Undeclared identifier aaaaaa where aaaaaa was INT, FLOAT or CHAR.

Cause INT, FLOAT and CHAR are predefined in the OS/2 and Windows NT compilers but are not predefined in the C89 compiler.

Solution We defined these explicitly with the following statements:
typedef int INT
typedef long FLOAT
typedef long CHAR

### 5.2.8 Too Many Statements in Source Code

Message ERROR CBC3191 ./osi.c:1059 The character - is not a valid C source character.

Code The line number in the error message was the last line in the source code. However, we got this message no matter what the content of that line was.

Cause This problem seems to result from too many pre-compiler directives or macros in the same source file.

Solution We removed unnecessary pre-compiler directives (those used for debugging purposes) and instanciated all macros in libraries where this problem existed. For one program, we also had to divide the source into smaller components to eliminate the error.

### 5.2.9 BYTE, WORD and DWORD Definitions

Message We got some compiler errors due to changes in the Notes C API release that we used.

Cause BYTE, WORD and DWORD were defined in the application code. These variable definitions were redundant, but were there to ensure that the variables were defined. The C89 compiler did not like those definitions because the variables are already defined in the Notes C API. This is documented in *Notes C API 4.51 Reference (api451re.nsf)*.

Solution Since we were redefining the variables to the values that we expected the Notes C API would want, we added precompiler directives to conditionally remove the definitions from our header file. The new code is shown in Figure 20 on page 33.

```
#ifndef OS390
    typedef unsigned char  BYTE;
    typedef unsigned short WORD;
    typedef unsigned long  DWORD;
#endif
```

*Figure 20. Removing the Definitions of BYTE, WORD and DWORD*

### 5.2.10 SEARCH_MATCH Structure

Message ERROR CBC3022 ./workflow.c:1123 "MatchesFormula" is not a member of "struct {...}"

Cause This is a documented change in the Notes C API between version 3 and version 4. See *Notes C API 4.51 User Guide (api451ug.nsf)*, document titled *Changes to Release 4.0*.

Solution We changed the SEARCH_MATCH structure as specified in the above document. The new code is shown in Figure 21.

```
#ifndef NOTES_V4
   if (search_info->MatchesFormula != SE_FMATCH )
#else
   if (!(search_info->SERetFlags & SE_FMATCH))
#endif
```

*Figure 21. SEARCH_MATCH Structure Change*

## 5.3 Linkedit Differences

Once we had eliminated all of the compiler error messages, the linker produced the messages in this section.

**Note:** It is a good idea to use the link options `-Wl,p,map` to produce a full pre-linker map. This allows you to map generated function names back to the real API name. The OS/390 linker uses seven character, upper case unique names, so this map will help you identify the real source of link failures.

### 5.3.1 STRICMP and STRNICMP Unresolved

Message IEW2456E 9207 SYMBOL STRICMP UNRESOLVED. MEMBER COULD NOT BE INCLUDED FROM THE DESIGNATED CALL LIBRARY.

IEW2456E 9207 SYMBOL STRNICMP UNRESOLVED. MEMBER COULD NOT BE INCLUDED FROM THE DESIGNATED CALL LIBRARY.

Cause The string functions `stricmp` and `strnicmp` are not supported by the C89 compiler.

Solution We wrote these two functions to have the same functions as those in the OS/2 and Windows NT compilers. These functions are only compiled if the platform is OS/390. The code John wrote is shown in Figure 22 on page 34.

```
#ifdef OS390
int stricmp(char * string1, char * string2){
   int i;
   for (i=0; i<strlen(string1) && i<strlen(string2) ;i++) {
     if (toupper(string1[i]) < toupper(string2[i])) return -1;
     if (toupper(string1[i]) > toupper(string2[i])) return 1;
   }
   if (strlen(string1) == strlen(string2))    return 0;
   if (strlen(string1) < strlen(string2)) return -1;
   return 1;
}

int strnicmp(char * string1, char * string2, int n){
   int i;
   for (i=0; i<n && i<strlen(string1) && i<strlen(string2) ;i++) {
     if (toupper(string1[i]) < toupper(string2[i])) return -1;
     if (toupper(string1[i]) > toupper(string2[i])) return 1;
   }
   if (strlen(string1) < n && strlen(string2) < n) {
     if (strlen(string1) == strlen(string2))    return 0;
     if (strlen(string1) < strlen(string2)) return -1;
     return 1;
   }
   if (strlen(string1) < n)
      return -1;
   if (strlen(string2) < n)
      return 1;
   return 0;
}
#endif
```

*Figure 22. Code for stricmp and strnicmp Functions*

### 5.3.2 INT Unresolved

Message IEW2456E 9207 SYMBOL INT UNRESOLVED. MEMBER COULD NOT BE INCLUDED FROM THE DESIGNATED CALL LIBRARY.

Cause. The Notes C API Toolkit assumes that INT is defined by the underlying operating system. However, INT is not a defined typedef in the C89 include library (see global.h).

Solution We added the following line to the existing typedefs:
typedef int INT;

### 5.3.3 @UNLINK Unresolved

Message IEW2456E 9207 SYMBOL @UNLINK UNRESOLVED. MEMBER COULD NOT BE INCLUDED FROM THE DESIGNATED CALL LIBRARY.

Cause _unlink is a Windows NT function that removes a connection to a file. This function is not supported on OS/390.

Solution We replaced the call to _unlink() with a call to the equivalent C89 function unlink() in the application code.

### 5.3.4 @STRDATE and @STRTIME Unresolved

Message IEW2456E 9207 SYMBOL @STRDATE UNRESOLVED. MEMBER COULD NOT BE INCLUDED FROM THE DESIGNATED CALL LIBRARY.

IEW2456E 9207 SYMBOL @STRTIME UNRESOLVED. MEMBER COULD NOT BE INCLUDED FROM THE DESIGNATED CALL LIBRARY.

Cause The string time and date functions `strdate` and `strtime` are not defined by either POSIX/XPG4 or ANSI, and the C89 compiler does not support them.

Solution We removed the calls to strdate and strtime since they were not important to the function of the code. If they were needed, we would have written equivalent functions using the `asctime()` function.

### 5.3.5 NOTESMAI Unresolved

Message IEW2456E 9207 SYMBOL NOTESMAI UNRESOLVED. MEMBER COULD NOT BE INCLUDED FROM THE DESIGNATED CALL LIBRARY.

Cause The link did not include `notesai0`. There are differences between the OS/2, Windows NT and UNIX platforms in the way applications have to be linked with Notes. These are described in the Notes V4 API reference documentation.

Solution We changed the way that we link to the Notes API. We removed the code which uses a PASCAL calling convention (which is not supported on UNIX) and replaced it with precompiler macro `LNPUBLIC`. This macro is provided in the Notes C API and it takes care of the correct calling convention so that the developer does not need to write different calls for different platforms. The code we changed was:

```
STATUS far PASCAL AddInMain(
   HMODULE  hModule,
   int      argc,
   char    *argv[]
)
```

We replaced that code with:

```
STATUS LNPUBLIC AddInMain(
   HMODULE  hModule,
   int      argc,
   char    *argv[]
)
```

As this program is structured as a Notes `AddInMain`, it must be linked with the bootstrap code `notes0.o`. and the addin0 code `notesai0.o`.. So we added these lines to our make file:
BOOTOBJS = $(LOTUS)/notesapi/lib/os390/notes0.o
BOOTOBJS = $(LOTUS)/notesapi/lib/os390/notesai0.o

### 5.3.6 Link Edit of Application Fails

At one stage, the link edit of an application failed. The code was compiled using -D__STRING_CODE_SET__="ISO8859-1" and _Ascii_a.h was included.

The error was that we had not modified the application's make file to specify libascii.a on the link edit.

### 5.3.7 Link Edit Shows libascii.a Is Not Found

Another problem that you may come across is that the link-edit shows libascii.a was not found.

Part of the installation of libascii includes compiling all the libascii code to create the libascii.a archive. Verify that the libascii.a archive has been created and that it is in a directory that the linkage editor can find.

# Chapter 6. Testing the Code

This chapter describes our experiences in testing the WIT code after compiling and linking it without error.

## 6.1 How We Ran WIT

We did much of our testing by running the WIT program from a Telnet session into the OS/390 UNIX environment. To start WIT, we typed in the executable file name `.wit32` at the UNIX command prompt. Some of the options on this command are:

**-v** List product version (used for testing)
**-f name** Pass the name of a Notes database

Normally WIT runs from the Domino server console and we also ran it that way. We followed the instructions in the Notes API documentation and copied the executable to our `$Notes_ExecDirectory` directory. Then we loaded it by issuing the Domino server command `load wit32 -v` with the appropriate options.

As an alternative to copying the WIT executable to `$Notes_ExecDirectory`, we also tested defining a UNIX symbolic link in `$Notes_ExecDirectory` that pointed to the WIT program in another directory. That also worked, and may be preferable for managing your system.

We tried to run WIT from OMVS but we were not successful. We did not do any problem determination as we had the Telnet method working. It may have been a setup problem, or it may be that the WIT application will not run in that environment because it has some restrictions that the Telnet environment does not have.

**Note:** Make sure that the executable file for the Notes API program has the same owner and group as the other Notes binaries and also has the set group_id bit set. See *Notes C API 4.51 User Guide (api451ug.nsf)* Chapter 3 “Platform Specifics” section “Building UNIX Applications.”

### 6.1.1 When WIT Did Not Run

During one of our tests of WIT, it would not run from the Telnet session. To check the environment, we reran the `intro` sample and it also did not run. We therefore had a problem with the Domino server environment.

To resolve the problem, we stopped the Domino server and then restarted it. WIT then ran again.

## 6.2 No Resource Compiler

In the OS/2 and Windows NT environments, the WIT application messages are defined in a resource file. The resource file is compiled with a resource compiler that is generally included in the C compiler and linker package. The application then uses an OSLoadString command to get the text for a message number. Since the message text is compiled separately from the application, it is possible to change the message text without recompiling the application.

In the OS/390 environment we do not have a resource compiler. We were aware of this at the time of compiling, but we got a clean compile and link without changing this area.

The lack of a resource compiler caused us problems in several areas:

- Message text was not displayed on the console. We discuss that in 6.2.1, “Symptoms of No Resource Compiler.”
- The Domino server status messages for WIT were missing. We discuss that in 6.2.4, “Changes for the Status Line” on page 40.
- The status messages when WIT starts were missing. We discuss that in 6.3, “Status Missing in WIT Startup” on page 42.

**Note:** After the project we discussed this further with OS/390 development. The *Notes C API 4.51 User Guide (api451ug.nsf)* mentions explicit support for .res files in only Windows and OS/2. Some UNIX platforms have a resource compiler, though it is not part of the Notes C API Toolkit. We believe that you can use another UNIX resource compiler to create .res files for OS/390, at least with the current releases.

## 6.2.1 Symptoms of No Resource Compiler

For the first run of the WIT program, we started it from a Telnet screen and got the results in Figure 23.

```
/usr/lpp/wit > wit32 -v
06/17/97 05:52:32 PM  33:0B
06/17/97 05:52:32 PM  33:0C
06/17/97 05:52:32 PM  33:00
06/17/97 05:52:32 PM  33:01
06/17/97 05:52:32 PM  33:02
06/17/97 05:52:32 PM  33:04
```

*Figure 23. Invalid Output as No Resource Compiler*

Each line should display some text after the timestamp. Instead we got message numbers displayed. This is because the message text is not being returned to the program.

We tried using the resource file (.res) generated by the OS/2 resource compiler. We used ftp to transfer the file to OS/390 in both binary and text mode. In both cases, when we started wit32 we got the message:

```
Corrupted resource file (bad resource length):        wit32.res
```

We therefore had to find another solution.

## 6.2.2 Creating Our Own Message Array

First we had to create our own message array on OS/390, rather than relying on the resource compiler to store that for us. We tried to maintain the messages in a format that could be used with the resource compiler on platforms that have one, or without it on other platforms. This new code is conditionally compiled on the OS/390 platform.

We created our own array to store the message numbers and corresponding text strings. The structure definition is shown in Figure 24 on page 39.

```
typedef struct _StatusRec         /* structure used to handle resource
                                     string table    */
{
    STATUS sError;
    char   szErrStr[256];
}   STATUSREC;
```

*Figure 24. Structure Definition for Messages*

We then initialized this array with the errorcodes and strings as shown in Figure 25. The application error codes begin with PKG_ADDIN+0, as defined by the Notes C API.

This message table is now compiled as a .h file and linkedited into the application code. This has the disadvantage that any changes to message text require a recompile of the application.

```
STATUSREC srecs[NUM_MSGS+1] =
{
IDS_WIT_RDATE,          "Release Date: 6/20/96",
IDS_WIT_PBY,            "Programmed & Designed By Application Partners, Inc.",
IDS_WIT_MESSAGE,               "WIT %s",
IDS_WIT_ACTERROR,       "WIT Error In Action: %s",
IDS_WIT_OUTPUT,                "%s",
IDS_WIT_DEBUG,                 "WIT Debug: %s",
IDS_WIT_ALLOCERR,       "Error Allocating Memory|||",
IDS_WIT_ERROR,          "WIT Error: %s",
IDS_WIT_TRACE,          "WIT Trace: %s",
IDS_WWF_CDOPEN,          "Could not open Control Doc",
     .
     .       /* here are defined all the other error codes and strings */
     .
}
```

*Figure 25. Initialize Structure with Message Text*

### 6.2.3 Program Logic Changes

Within WIT, all messages are processed in a single function. Therefore only one function needed to be changed. The original function contained the code in Figure 26.

```
  OSLoadString( NULLHANDLE, msg_num, message, FORMULA_STR_LENGTH );
  OSTranslate( OS_TRANSLATE_LMBCS_TO_NATIVE, message, strlen(message),
               message1,256 );
```

*Figure 26. Original Code to Handle Messages*

The first line calls the Notes C API function OSLoadString and passes it the input variable msg_num. For Notes API message numbers, the message text is returned. For user messages, the message text is retrieved from the message table linked using the resource compiler. The message text is returned to the

program in the variable `message`. The second line then translates the message text from Lotus multi-byte character set (LMBCS) to the native character set.

#### 6.2.3.1 New Code

John changed the generic message processing function, as shown in Figure 27.

```
#ifndef OS390
  OSLoadString( NULLHANDLE, msg_num, message, FORMULA_STR_LENGTH );
  OSTranslate( OS_TRANSLATE_LMBCS_TO_NATIVE, message, (WORD)strlen(message),
               message1,256 );
#else
  if(msg_num >= FIRST_MESS && msg_num < LAST_MESS) {
    strcpy( message1, srecs[msg_num-FIRST_MESS+1].szErrStr);
   }
  else {
    AddInLogMessageText("Lotus Error Message",msg_num);
   }
#endif
```

*Figure 27. New Code to Handle Messages*

Conditional code has been added for the OS/390 platform.

For platforms other than OS/390, the code is the same as before, except that (`WORD`) has been added in the `OSTranslate` line. This was to fix a warning that John was getting from the OS/2 compiler. It is not related to this porting exercise; John had previously ignored the message but he chose to fix it now.

On OS/390, we first check whether this is a user message number or a Notes C API message number. If it is a user message number, we use the C string copy function `strcpy` to copy the message from the array of messages that we defined and initialized earlier. If it is a Notes C API message number, we call the Notes C API function `AddInLogMessageText` which writes a message to the console and the log.

We found the `AddInLogMessageText` function useful to write out progress and error messages to the log during various stages of debugging.

### 6.2.4 Changes for the Status Line

Since we do not have a resource compiler, we also need to change the way in which WIT provides status messages to Notes.

#### 6.2.4.1 Initialization Section

In the initialization section of the main function, we used the conditional code in Figure 28 on page 41. This sets the text from our message array rather than from the compiled resources.

```
#ifndef OS390
   AddInSetStatus( IDS_WWF_STARTMESS );
   AddInLogMsg(    IDS_WWF_STARTMESS, NULL );
   AddInLogMsg(    IDS_WWF_STARTEXE, NULL );
#else
   AddInSetStatusText( srecs[IDS_WWF_STARTMESS-FIRST_MESS+1].szErrStr );
   AddInLogMessageText( srecs[IDS_WWF_STARTMESS-FIRST_MESS+1].szErrStr, NOERROR);
   AddInLogMessageText( srecs[IDS_WWF_STARTEXE-FIRST_MESS+1].szErrStr, NOERROR);
#endif
```

*Figure 28. Initialization Section*

#### 6.2.4.2 Setting the Status Line

As needed throughout the main function, we set the status from our message array, as shown in Figure 29.

```
#ifndef OS390
   AddInSetStatus( IDS_WIT_OUTPUT, error_str );
#else
   AddInSetStatusText( srecs[IDS_WIT_OUTPUT].szErrStr, error_str );
#endif
```

*Figure 29. Setting the Status Line*

#### 6.2.4.3 Termination Section

In the termination section of the main function, we used the code in Figure 30.

```
#ifndef OS390
   AddInSetStatus( IDS_WWF_TERMMESS );
   return(NOERROR)
#else
   AddInSetStatusText( srecs[IDS_WWF_TERMMESS-FIRST_MESS+1].szErrStr );
#endif
```

*Figure 30. Termination Section*

#### 6.2.4.4 Status Line

After making the changes discussed, we got the status line shown in Figure 31 on page 42.

```
# load wit -fwit/witdata1

# show tasks
    ...
      Task                Description

 Database Server      Perform console commands
 Database Server      Listen for connect requests on TCPIP
 Database Server      Idle task
 Database Server      Idle task
 Database Server      Server for John LaFata/itso390b on TCPIP
 /usr/lpp/lotus/notes Idle in wit/witdata1
 Schedule Manager     Idle
 Admin Process        Idle
 Agent Manager        Idle
 Indexer              Idle
 Router               Idle
 Replicator           Idle
 Billing              Idle
```

*Figure 31. WIT Status Line*

**Note:** On OS/2 and Windows, the task name is the first 20 characters of the program name with the path name removed. On OS/390, we get the first 20 characters including the path name. This may be consistent with other UNIX platforms. See *Notes C API 4.51 User Guide (api451ug.nsf)* chapter 4-4 *Server Add_In Tasks* section *AddInSetStatus, AddInSetStatusText, AddInSetStatusLine*.

## 6.3 Status Missing in WIT Startup

We had another problem related to the lack of a resource compiler. When we started WIT, we got the messages in Figure 32 on the screen that we started WIT from:

```
/usr/lpp/wit > wit32 -v
06/18/97 06:52:03 PM  WIT(tm), The API Library(tm) V3.2.9.1 Copyright (c)
1992-1996 Application Partners, Inc.  All Rights Reserved Worldwide.
06/18/97 06:52:03 PM  WIT: Starting ...
06/18/97 06:52:03 PM  33:00
06/18/97 06:52:03 PM  33:01
06/18/97 06:52:03 PM  33:02
   ...
```

*Figure 32. WIT Startup with Three Missing Status Messages*

The messages with times 33:00, 33:01 and 33:02 should show information about the WIT program.

We then made the code changes in 6.2.4, “Changes for the Status Line” on page 40 and got the messages in Figure 33 on page 43.

```
/usr/lpp/wit > wit32 -v
06/18/97 09:22:19 PM  WIT(tm), The API Library(tm) V3.2.9.1 Copyright (c)
1992-1996 Application Partners, Inc.  All Rights Reserved Worldwide.
06/18/97 09:22:19 PM  WIT: Starting ...
06/18/97 09:22:19 PM
06/18/97 09:22:19 PM  Version 3.2.9.1
06/18/97 09:22:19 PM  Release Date: 6/20/96
   ...
```

*Figure 33. WIT Startup with One Missing Status Message*

We now had the WIT version number and release date but we were still missing the program name. This is stored in the 0th element in the string array used for messages. To see if the problem was related to that, John made the 0th element blank and moved all the other elements forward by one. He then changed the code that indexes into the array to increment by one, for example:

```
AddInSetStatusText( srecs[IDS_WWF_TERMMESS-FIRST_MESS].szErrStr );
```

was changed to

```
AddInSetStatusText( srecs[IDS_WWF_TERMMESS-FIRST_MESS+1].szErrStr );
```

This solved the problem and we got the full set of status messages as shown in Figure 34.

```
/usr/lpp/wit > wit32 -v
09/03/97 12:06:07 PM  WIT(tm), The API Library(tm) V3.2.9.1 Copyright (c)
1992-1996 Application Partners, Inc.  All Rights Reserved Worldwide.
09/03/97 12:06:07 PM  WIT: Starting ...
09/03/97 12:06:07 PM  WIT, Version 3.2.9.a
09/03/97 12:06:07 PM  Version 3.2.9.1
09/03/97 12:06:07 PM  Release Date: 6/20/96
   ...
```

*Figure 34. WIT Startup with All Status Message*

## 6.4 Message Strings Corrupted

One problem that we found was that the messages produced by the WIT program were corrupted. When we ran WIT, we got the output shown in Figure 35.

```
/usr/lpp/wit > wit32 -v
06/18/97 09:22:19 PM  WIT(tm), The API Library(tm) V3.2.9.1 Copyright (c)
1992-1996 Application Partners, Inc.  All Rights Reserved Worldwide.
06/18/97 09:22:19 PM  WIT: Starting ...
06/18/97 09:22:19 PM
06/18/97 09:22:19 PM  Version 3.2.9.1
06/18/97 09:22:19 PM  Release Date: 6/20/96
  .
06/18/97 09:22:19 PM  Determined msg_error_eve: -1
  .
06/18/97 09:22:19 PM  MESSAGE from DB %s - %snating
```

*Figure 35. Message Strings Corrupted*

You see in the messages that:

- The message `Determined msg_error_level: -1` is being written as `Determined msg_error_eve: -1`. The `l` characters have been removed.
- The message `MESSAGE from DB UNKNOWN - WIT terminating` is being written as `MESSAGE from DB %s - %snating`.

The module that writes these messages uses the functions `strcpy`, `memcpy` and `sprintf` to manipulate strings. These are all functions that are included in libascii.

After some time, we realized that we had neglected to add the include file `_Ascii_a.h` to the sources file for this program. The messages were therefore not being converted to EBCDIC correctly. After adding the include file, the messages were displayed correctly.

### 6.4.1 Printf Output to Stdout Is Unreadable

In another test we found that the printf output to stdout was unreadable. The code was compiled using `-D__STRING_CODE_SET__="ISO8859-1"` and _Ascii_a.h was included.

The error was that some header files were missing. For example, libascii requires `<stdio.h>` to be included to use the libascii version of printf(). If your application uses ctime() and you wish to use the libascii ctime, then `<time.h>` must be included.

Another possible error is that _Ascii_a.h must be included *after* all the other header files.

## 6.5 Difference in Database Name Specification

In our testing, we found that database names need to be specified differently on S/390:

- The names are case-sensitive, unlike on OS/2 and Windows.
- You must specify `.nsf` in the database name, while it is assumed on OS/2 and Windows.
- In the path name, the slash goes in the opposite direction: / on UNIX servers, \ on PC servers.

We found this out when we ran WIT with a command option that opens a database. The command to start WIT was: `wit32 -f wittest`. This should open the Notes database wittest.nsf. However, when we issued this command, we got a reply that `File does not exist`.

John then tried the same command with the database suffix `.NSF` and the database name in uppercase. The command he used was

```
wit32 -fWITTEST.NSF -d0 -o >out
```

This time the database was opened and the program produced other messages before failing with another error.

### 6.5.1 Coding for Different Path Separator

Within your application you need to use the correct path separator for the platform you will run on. One way to do that is to define a variable PATHSEP to be the path separator character, as shown in Figure 36. Then use the variable PATHSEP to build your file name strings.

```
#ifdef UNIX
 #define PATHSEP "/"
#else
 #define PATHSEP "\"
#endif
```

*Figure 36. Path Separator for Different Platforms*

### 6.5.2 Caching of Database Names

The WIT program code made the assumption that you could refer to the database WITDATA.NSF with any of these names:

WITDATA.NSF
WITDATA.nsf
witdata.NSF
witdata.nsf

They were assumed to all refer to the same database, which is true on the Intel platforms where all file names are in uppercase. However, on UNIX platforms, file names are case-sensitive so, if you referred to the above four files on a UNIX platform, you would be referring to four different files.

John changed the code to take account of case-sensitive database names on UNIX.

## 6.6 Different Sizes for Handles

The next problem that we met is shown in Figure 37 on page 46. Note that by this time John had added some printf statements into the WIT code to print out variables and assist in debugging. Note also that some of the messages were not printing correctly. At this stage, we still had not fixed the problem discussed in 6.4, “Message Strings Corrupted” on page 43.

```
/usr/lpp/wit > wit32 -fWITTEST.NSF -d0 -o >out
06/19/97 12:17:24 PM  WIT(tm), The API Library(tm) V3.2.9.1 Copyright (c)
1992-1996 Application Partners, Inc.  All Rights Reserved Worldwide.
06/19/97 12:17:24 PM  WIT: Starting ...
06/19/97 12:17:25 PM  Our message number
06/19/97 12:17:25 PM  Retrieved and translated message to message1
06/19/97 12:17:25 PM  Reading The WIT Information
06/19/97 12:17:25 PM  Constructed message in New format
06/19/97 12:17:25 PM  DB WITTEST.NSF whie running ./readinfo.c ine 1414 xxx DB
%s whie running %s ine %d xxx %s
06/19/97 12:17:25 PM  STATUS from DB %s whie running %s ine %d xxx %s

CEE3206S The system detected a specification exception.
         From entry point OSLockReadFRWSem at compile unit offset +0004BF0A at
address 089E8C8A.
[1] + Done(132) wit32 -fWITTEST.NSF -d0 -o >out
  1694498847    FSUM7739 Illegal instruction    ./wit32
```

*Figure 37. Specification Exception - 1*

This proved to be a difficult problem to track down. John added printf statements to the code to display the variables being used and eventually we saw the messages in Figure 38.

```
/usr/lpp/wit > wit32 -fwittest.nsf -d0 -o >out
06/19/97 03:51:09 PM  WIT(tm), The API Library(tm) V3.2.9.1 Copyright (c)
1992-1996 Application Partners, Inc.  All Rights Reserved Worldwide.
06/19/97 03:51:09 PM  WIT: Starting ...
06/19/97 03:51:09 PM  Our message number
06/19/97 03:51:09 PM  Retrieved and translated message to message1
06/19/97 03:51:09 PM  Reading The WIT Information
06/19/97 03:51:09 PM  Constructed message in New format
06/19/97 03:51:09 PM  DB WITTEST.NSF while running ./readinfo.c line 1418 xxx
Reading The WIT Information
06/19/97 03:51:09 PM  STATUS from DB WITTEST.NSF while running ./readinfo.c
line 1418 xxx Reading The WIT Information
06/19/97 03:51:09 PM   After NSFDbOpen..
06/19/97 03:51:09 PM  BEGIN OBJECT CALL
06/19/97 03:51:09 PM   After Create Array...
06/19/97 03:51:09 PM   After Lock Array...
06/19/97 03:51:09 PM   After strupr...
06/19/97 03:51:09 PM   Before NIFFindView...
06/19/97 03:51:10 PM   After NIFFindView...
06/19/97 03:51:10 PM   After NIFOpenCollection...
06/19/97 03:51:10 PM  size of Handle: 4, size of HCOLLECTION: 2
CEE3206S The system detected a specification exception.
         From entry point OSLockReadFRWSem at compile unit offset +0004BF0A at
address 089EDC8A.
[1] + Done(132) wit32 -fwittest.nsf -d0 -o >out
  1241514020    FSUM7739 Illegal instruction    ./wit32
```

*Figure 38. Specification Exception - 2*

This code works on OS/2 and Windows NT. John therefore compiled the current version of the code on OS/2 and ran it. The highlighted line in Figure 38 came out on OS/2 as:

```
06/19/97 03:51:10 PM  size of Handle: 2, size of HCOLLECTION: 2
```

The WIT code assumes that Handle and HCOLLECTION are the same size. This is true on OS/2 and Windows, but it not true on UNIX servers.

**Note:** If the WIT code already ran on other UNIX servers, this difference would already have been coded for.

### 6.6.1 Handle Sizes

The Notes C API include library `/usr/lpp/lotus/notesapi/include/` contains the handle definitions shown in Table 2.

*Table 2. Notes C API Handle Definitions*

| File name | Definition |
|---|---|
| nif.h | typedef WORD HCOLLECTION; /* Handle to NIF collection */ |
| global.h | #if defined(DOSW16) \|\| defined(OS390)<br>typedef unsigned int HANDLE;<br>#else<br>typedef unsigned short HANDLE;<br>#endif |
| nsfdata.h | #define DBHANDLE HANDLE |

**Notes:**

1. HCOLLECTION is a word, which is 2 bytes on all platforms.
2. HANDLE, and therefore DBHANDLE, are unsigned short (2 bytes) on all platforms except DOSW16 and OS390. On these platforms they are unsigned int, which is 2 bytes on DOSW16 and 4 bytes on OS390.

Therefore, on OS/2 and Windows NT (Intel) platforms, database handles DBHANDLE and collection handles HCOLLECTION have the same size of two bytes. On OS/390 HCOLLECTION has a size of two bytes but DBHANDLE has a size of four bytes.

If you want your code to be portable, you should not make assumptions about the size of objects.

### 6.6.2 Solution

John changed WIT so that it did not assume that HANDLE and HCOLLECTION are the same size. The original line in the code was:

```
NIFUpdateCollection( ( HANDLE ) *objHandle );
```

John changed it to:

```
NIFUpdateCollection( ( HCOLLECTION ) *objHandle );
```

**Note:** An alternative approach may have been to make HCOLLECTION and HANDLE the same size, by making HCOLLECTION reference the definition of HANDLE in global.h.

## 6.7 Rich Text Format

After addressing the previous problem, we reran the test and got the DocLink problem shown in Figure 39.

```
   .
06/19/97 07:09:09 PM  Lotus message number: Entry not found in index
06/19/97 07:09:09 PM  ERROR from DB WITTEST.NSF while running ./readinfo.c line
694 xxx Illegal DocLink to Target DocInfo on Action #: 0
```

*Figure 39. Doclink Problem*

This happened because the application tried to open a document with a DocLink using *host format rich text*. Not all platforms support this function.

### 6.7.1 Rich Text Formats

The original way to access rich text was in host format rich text. This was designed for the Intel file system. You can use pointer arithmetic to move through the rich text structure.

When Lotus ported Notes to UNIX platforms, they found that this no longer worked. You cannot rely on byte arithmetic because of differences in the operating system. Therefore Lotus introduced canonical rich text format, which works at a higher level. Canonical rich text format works on all platforms and is the recommended way to access rich text on all platforms.

### 6.7.2 Solution

This was a difference that John was aware of before we started the port. It is documented in *Notes C API 4.51 User Guide (api451ug.nsf)* Chapter 12-1 *Notes Canonical Format*. John changed the complete application to use canonical rich text format. This was the hardest task in the entire port.

**Note:** If the code had already been written to use canonical rich text format, then no changes would have been required in the port to OS/390.

## 6.8 Segmentation Violation When Close Domino Server

During our testing we got an intermittent segmentation error. This occurred when we closed down the Domino server. The error message is shown in Figure 40.

```
From entry point ExecDeleteTask at compile unit offset -F9150856 at
    address 024610CE.
[1] + Done(139) server
      67108886    FSUM7746 Segmentation violation /usr/lpp/lotus/notes/
          latest/os390/server
```

*Figure 40. Segmentation Violation When Close Domino Server*

This error did not occur on OS/2 or Windows NT. It typically means that memory is not being deallocated correctly.

John decided to check for unmatched memory allocation and de-allocation. He put in a trace of the allocation and deallocation functions in WIT and matched them together. John found a section of code that was conditionally removed for the port. However, the allocation code that supported it had *not* been conditionally removed. When that code was removed, the segmentation violation error did not occur again.

**Note:** We have seen examples in other projects where OS/390 is much more stringent than other platforms in its control of memory use by applications. It does this to protect other applications, and the operating system, from errors in memory use. On OS/390, therefore, you can see errors in memory use that are not tracked and highlighted on other, less stringent, platforms.

## 6.9 Restart after a Server Protection Exception

We ran our project with beta code for the Domino server, Notes C API Toolkit and libascii. At one stage we got a server protection exception as shown in Figure 41.

```
/usr/lpp/lotus/notesapi/samples/basic/intro > intro
CEE3204S The system detected a protection exception.
         From compile unit ./ospanic.c at entry point Panic at statement 327 at
compile unit offset +00000410 at address
          088B4BF8.
[1] + Done(139) intro
  117440532     FSUM7746 Segmentation violation ./intro
/usr/lpp/lotus/notesapi/samples/basic/intro >
```

*Figure 41. Server Protection Exception*

After this the Domino server console no longer responded.

The procedure to restart the server is documented in *Lotus Domino Server Release 4.5 on S/390: Installation, Customization and Administration*, SG24-2083. We give a brief review of the steps in Figure 42.

```
Cntl+Break                  <==  Remove the server console from Telnet

Logon to Telnet again with same user ID

cd /usr/lpp/lotus/bin/tools
killnotes                   <==  Remove processes, shared memory and
                                 semaphores

cd /notesdata               <==  Change to Domino server data directory

/usr/lpp/lotus/bin/server   <==  Start the Domino server
```

*Figure 42. Restart the Domino Server*

# Appendix A. API 4.5.1 Code Readme File

This is a copy of the readme.390 file for the Lotus Notes C API 4.5.1 Toolkit as we downloaded it from the Web on December 5, 1997. It can be downloaded from the IBM Web site: http://www.s390.ibm.com/products/domino or from the Lotus Web site: http://193.164.160.162/ldw.nsf/Data/Document1289.

This readme file is contained in the toolkit tar file. On our system it was unpacked to the file /usr/lpp/lotus/notesapi/readme.390.

This file is © 1996, 1997 Lotus Development Corporation. Used with permission of Lotus Development Corporation. Lotus Notes is a registered trademark and Domino is a trademark of Lotus Development Corporation. It is understood that whenever possible, full credit will be given to Lotus Development Corporation.

```
(c) 1996, 1997 Lotus Development Corporation.  All rights reserved.
This software is subject to the Lotus Notes C API Software Development
Kit Agreement (in the apilicns.txt file in the notesapi directory of
this toolkit), Restricted Rights for U.S. government users, and
applicable export regulations.

*****************************************************************************

                        Lotus Notes C API Release 4.51

                        Installation Instructions for
                                IBM OS/390

*****************************************************************************

This is the Lotus Notes C API Toolkit Release 4.51.


Platform Information:
--------------------

Lotus Notes C API (Application Program Interface) Toolkit Release 4.51
supports development of API programs for for Notes Release 4.51 on
multiple platforms, including Microsoft Windows, IBM OS/2, Novell
NetWare, Macintosh, IBM AIX, IBM OS/390, HP-UX, Sun SPARC Solaris and
Sun x86 Solaris.

This document explains how to install the Lotus Notes C API Release 4.51
on an IBM OS/390 running the Lotus Domino Server.

Lotus Notes C API Toolkit Release 4.51 supports developing programs for
IBM OS/2, Microsoft Windows, and Novell NetWare.  For information on
these configurations and installation information, please see the file,
readme.pc, in the PC Lotus Notes C API Toolkit directory,
lapi_pc\notesapi.

Lotus Notes C API Toolkit Release 4.51 supports developing programs for
the UNIX platforms IBM AIX, Sun SPARC Solaris, Sun x86 Solaris, and
HP-UX.  For information on these configurations and installation
information, please see the file, readme.unx, in the UNIX Lotus Notes C
API Toolkit directory, lapi_unx\notesapi.
```

Lotus Notes C API Toolkit Release 4.51 supports developing programs for Apple Macintosh computers. For information on Macintosh configurations and installation information, please see the file, readme.mac, in the Macintosh Lotus Notes C API Toolkit directory, lapi_mac\notesapi.

Installation Instructions:
---

This section explains how to install the Lotus Notes C API Toolkit Release 4.51 on an IBM OS/390 system running the Lotus Domino Server.

Lotus Notes C API Toolkit Release 4.51 supports developing programs for for IBM OS/390 Release 3 with the OpenEdition Shell and Utilities feature using the IBM OS/390 C/C++ Language Environment, compiler, and tools.

The OS/390 platform with OpenEdition is very similar to other UNIX platforms (such as IBM AIX), and this toolkit will be familiar to UNIX users. One major difference is that the character code-set for OS/390 is EBCDIC, while on most other platforms, it is ASCII. To make it easier to port ASCII-based C/C++ applications (such as Lotus Notes applications) to OS/390, a new compiler predefined macro and a conversion package, "libascii", are provided.

A new OS/390 V1R3.0 C/C++ compiler predefined macro, __STRING_CODE_SET__="ISO8859-1", will generate ASCII characters rather than the default EBCDIC characters. Using this macro with the libascii conversion package will provide an ASCII-like environment which may make it easier to port your application to OS/390.

The libascii package can make ASCII-based C/C++ applications easier to develop in the OpenEdition environment by providing an ASCII interface layer for some of the more commonly used C/C++ run-time functions. The libascii package can be downloaded from:

        http://www.s390.ibm.com/oe/libascii.html

Follow the instructions for downloading libascii and compiling the libascii code. See the "readme" file within the libascii directory (created during installation) for the latest information on the package. Be sure to set the LIBASCII_PATH environment variable to point to the package location.

Most of the sample programs provided by this toolkit rely on both the __STRING_CODE_SET__ compiler macro and the libascii package. A few can be made to operate in either an ASCII environment or in EBCDIC. The EBCDIC-based samples are identified by the files "os390.mak" which contain the CODE_SET variable as follows:

```
    # Set CODE_SET to 'ascii' to compile with ASCII format data.  Any
    # other value (such as 'ebcdic') will compile with the OS/390 native
    # EBCDIC format.  This sample program supports either code set.
    CODE_SET = ascii
    #CODE_SET = ebcdic
```

Installation:
---

1. Copy the file "api451.tar.Z" to a directory from which development will originate, such as "/usr/lpp/lotus". If you use FTP or other file transfer applications, be sure the transfer occurs in binary mode. Note that the sample applications will require access to the Lotus Domino Server execution directory, typically "/usr/lpp/lotus/notes/latest/os390".

2. From the development directory, unpack the "api451.tar.Z" file with the command:

   ```
   tar -xzvof api451.tar.Z
   ```

   The sub-directory "notesapi" will be created. It contains the Lotus Notes C API Toolkit.

3. Download and build the libascii package as per the instructions on the web location listed above. Ensure the environment variable "LIBASCII_PATH" points to the package location, for example:

   ```
   export LIBASCII_PATH=/usr/libascii
   ```

   To insure the API toolkit can successfully locate dependencies from the libascii package, issue the following commands:

   ```
   whence $LIBASCII_PATH/_Ascii_a.h
   whence $LIBASCII_PATH/libascii.a
   ```

   The result from each of the above should be the fully-qualified path and file name of the specified part.

4. Create the environment variable "LOTUS" to point to the development base from which the "notesapi" directory can be located:

   ```
   export LOTUS=/usr/lpp/lotus
   ```

   The make files for each sample application depend on this setting to locate the API library.

5. Ensure that the environment variable "Notes_ExecDirectory" points to the Lotus Domino Server execution directory:

   ```
   export Notes_ExecDirectory=/usr/lpp/lotus/notes/latest/os390
   ```

   The make files for each sample application depend on this setting to locate the libnotes DLL and side-deck. The Lotus Domino Server also depends on this setting. The execution directory should also be part of both the PATH and LIBPATH environment variable strings.

6. Ensure the file "notes.ini" is contained within a directory in the PATH environment variable. The Lotus Domino server depends on the notes.ini file located through the PATH for various functions, including locating the Notes data directory for database resolution. This same function is invoked through API interfaces, such as "NSFDbOpen()". Therefore, the environment from which C API programs are run must be similar to the environment from which the Lotus Domino Server is started.

   When the server opens the notes.ini file, it does so with READ/WRITE

authority. Therefore, the user running C API programs must have READ/WRITE capability for "notes.ini". This is likely not the default authority, as the group permission for notes.ini is READ only. Rather than change ownership or group permissions on the notes.ini installed for the server, the C API programmer can copy notes.ini and insure the copy appears first in the PATH environment variable. Note that you do not need to copy any other files from the Notes data directory; notes.ini specifies the data directory and the copy will continue to point to the original data directory.

7. Copy the Notes C API database documents to a location where they can be viewed, either to the server data directory or directly to your Notes client workstation. The files are:

```
Notes C API 4.51 User Guide      (api451ug.nsf)
Notes C API 4.51 Reference       (api451re.nsf)
```

## Sample Application build and execution

The Notes C API 4.51 User Guide contains specific instructions for each sample application provided. OS/390 supports a subset of the samples, specifically, those which execute in the server environment.

To compile and link a sample application, use the make utility against the OS/390-specific make file. For example, to build the intro sample:

```
cd /usr/lpp/lotus/notesapi/samples/basic/intro
make -f os390.mak
```

The result will be the program "intro". The intro sample does not require the Lotus Domino Server program to be started (other samples do). It does require that the database "intro.nsf" be available in the Domino server data directory. If the data directory is "/notesdata" (the default during server installation), copy all the provided sample databases to the data directory:

```
cd /usr/lpp/lotus/notesapi/notedata
cp * /notesdata/.
```

Now "intro" can be run:

```
/usr/lpp/lotus/notesapi/samples/basic/intro/intro
```

resulting in:

```
Lotus Notes C API Release 4.51 Sample Application


The title for the database intro.nsf is...

API Test Database (intro)
```

Note that the intro sample also supports an EBCDIC version which can produce the same result without the use of the libascii package. Edit the make file /usr/lpp/lotus/notesapi/samples/basic/intro/os390.mak and change the value of the CODE_SET variable to:

```
        CODE_SET = ebcdic
```

Remove objects from any previous compile, make, then run the intro sample again:

```
        cd /usr/lpp/lotus/notesapi/samples/basic/intro
        rm intro intro.o
        make -f os390.mak
        intro
```

Other samples are built similarly. Each sample directory contains a "readme.txt" for simple documentation. Those samples that are built as addin-servers (such as "addin") or as DLL's that are called from the server (such as "libdbdbdrv") must be accessible from the Notes execution directory. Therefore, once the program or DLL is built, copy it to $Notes_ExecDirectory prior to starting the Lotus Domino server.

# Appendix B. OS/390 OpenEdition libascii Read File

This is a copy of the readme file shipped with the libascii package. It was downloaded from the Web on December 5, 1997. The URL is:

```
http://www.s390.ibm.com/products/domino
```

If your browser does not work with that URL, use the following URL instead:

```
ftp://ftp.s390.ibm.com/u/ftp/os390/oe/toys/libascii.tar.Z
```

This file is contained in the package tar file. On our system it was unpacked to the file `/usr/libascii/readme`. For the latest information, use the version of the readme file in the package that you download.

To report problems or ask questions on libascii, send e-mail to: `libascii@vnet.ibm.com`.

```
************************************************************************
*                                                                      *
* libascii - ascii-ebcidic interface layer - README file               *
*                                                                      *
* Version 1.1.5                                                        *
*                                                                      *
* To report problems or ask questions send e-mail to:                  *
*                                                                      *
*             libascii@nvet.ibm.com                                    *
*                                                                      *
* Copyright:   Licensed Materials - Property of IBM.                   *
*              (C) Copyright IBM Corp. 1997.                           *
*              All rights reserved.                                    *
*                                                                      *
* License information:                                                 *
*                                                                      *
*   The libascii source code is provided free of charge and            *
*   may be distributed freely.  No fee may be charged if you           *
*   distribute the libascii source code (except for such things        *
*   as the price of a disk or tape, postage ).  The libascii           *
*   makefile will compile and produce a libascii.a archive file.       *
*   The libascii.a archive may be link edited  with any software       *
*   vendor  product.  Any software vendor product that is link         *
*   edit with libascii.a archive is free to distribute and charge      *
*   for that product.                                                  *
*                                                                      *
*   THIS PROGRAM IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY          *
*   KIND, EXPRESS  OR IMPLIED, INCLUDING THE IMPLIED  WARRANTIES       *
*   OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.           *
*   IBM does not warrant uninterrupted or error free operation of      *
*   the  Program, or that the Program is free from claims by a         *
*   third party of copyright, patent, trademark, trade secret,         *
*   or any other intellectual property infringement. IBM  has          *
*   no obligation to provide service, defect correction, or any        *
*   maintenance for the Program. IBM  has  no  obligation to           *
*   supply any Program updates or enhancements to you even if          *
*   such are or later become available.                                *
*                                                                      *
*    Under no circumstances is IBM liable for any of the               *
*    following:                                                        *
*                                                                      *
```

```
*      1.  third-party claims against you for losses or damages;    *
*      2.  loss of, or damage to, your records or data; or          *
*      3.  direct  damages,  lost  profits, lost savings,           *
*          incidental, special, or indirect damages or other        *
*          consequential damages,  even if IBM or its authorized    *
*          supplier, has been advised of the possibility of         *
*          such damages.                                            *
*                                                                   *
*    Some jurisdictions do not allow these limitations or           *
*    exclusions, so they may not apply to you.                      *
*                                                                   *
*                                                                   *
*********************************************************************
```

OVERVIEW

The libascii package helps you port ASCII-based C applications to the EBCDIC-based OS/390 OpenEdition environment.

If you are porting an application which has a lot of ASCII dependencies, then you may find there are a large number of changes required to port the application to EBCDIC-based OS/390.

The C/C++ Run-Time Library functions support EBCDIC characters. The libascii package provides an ascii interface layer for some of the more commonly used C/C++ Run-Time Library functions. libascii supports ascii input and output characters by performing the necessary iconv() translations before and after invoking the C/C++ Run-Time Library functions. Note that not all C functions are supported (see Limitations for additional information).

The OS/390 1.3.0 C/C++ compiler predefined macro __STRING_CODE_SET__="ISO8859-1", will generate ASCII characters rather than the default EBCDIC characters. Using this with libascii will provide an ASCII like environment.

The libascii package is as thread safe as the Run-Time library except where stated under limitations below.

Note that not all C functions are supported (see LIMITATIONS section for additional information).

In order to use the libascii functions you need to:

- Install the libascii code.
- Build the libascii.a archive file.
- Make minor modifications to your C source code and recompile, using the __STRING_CODE_SET__="ISO8859-1" flag
- Link edit your application with the libascii.a archive file.

The libascii archive also contains four functions to convert between IEEE floating point and S390 hex floating point.

INSTALLING THE LIBASCII CODE

```
     The libascii TAR file contains all the parts required to create
     the libascii archive. To install:

          1.Create a directory where you want libascii installed.

          2.Copy the TAR file to that directory.

          3.Unwind the file using this command:

               tar -xvfz libascii.tar.Z

     The libascii source files, makefile, and readme file should
     now be installed.

BUILDING THE LIBASCII ARCHIVE FILE (libascii.a)

     You can use the makefile file provided to build libascii.a
     from the libascii source files.

     To build libascii.a, cd into the directory with the libascii
     source files and issue the make command.

BUILDING and RUNNING SAMPLES

     After you have made the libascii.a archive file, cd into the
     samples directory and issue the make command. This builds the
     samples.

     To run the sample programs issue the following from the samples
     directory:

          sample1
          sample2

USING LIBASCII

     After you have installed libascii and built the archive file,
     to use the libascii functions you need to:

     o   Make minor modifications to your C source code and
         recompile, using the __STRING_CODE_SET__="ISO8859-1"
         predefined macro.

     o   Link edit your application with the libascii.a archive file.

     First it must determined which parts in your application will be
     compiled with -D__STRING_CODE_SET__="ISO8859-1" specified to
         generate ASCII strings.  It is possible that part of the application
     will be compiled to generate ASCII strings and the other part will
     be compiled to generate EBCDIC strings.

     For all parts compiled with the -D__STRING_CODE_SET__="ISO8859-1"
     libascii should be used.  The following steps describes how
     to use libascii:

     o   Find all the C/C++ Run Time Library functions which require
         EBCDIC input (for example, fopen()) or produce EBCDIC output
         (for example, readdir()). Both the "nm" shell command and
```

the following grep command will help perform this task. The following grep command will display the file names that contain a libascii function followed by the character '(' .

```
grep -l -f /the-libascii-directory/libascii.lst *.c
```

- Determine which C/C++ Run Time Library functions obtained above are not supported by libascii. (See LIMITATIONS section for additional information.) For any unsupported functions you will need to either add functions to libascii or change the application to add iconv() ASCII/EBCDIC conversions around the run time library calls.

- Make sure all the required header files are included as specified in the OS/390 C/C++ Run-Time Library Reference. For example strncasecmp() requires <strings.h> to be included.

- Add the following statement:

```
#include "_Ascii_a.h"
```

  to your source file, after all the existing #include statements.

- Recompile using the -D__STRING_CODE_SET__="ISO8859-1" option to cause the compiler to generate all strings defined in your program in ASCII rather than EBCDIC format.

- Link-edit your application with the libascii.a archive file.

FLOATING POINT CONVERSION

The libascii archive also contains four functions to convert between IEEE floating point and S/390 native hex floating point. The OS/390 1.3.0 C/C++ compiler does not support IEEE floating point. Math operators such as + (add) and / (divide) and run-time library functions such as printf() and sin() using IEEE floating point numbers will produce undefined results. The main difference between the two floating point formats is IEEE supports a larger range of numbers, up to 10 to the 308 power. S/390 supports better precision but a smaller range, up to 10 to the 75 power.

The four floating point conversion routines included in libascii are as follows:

```
void ConvertFloatToIEEE(void *source, void *destination);
void ConvertDoubleToIEEE(void *source, void *destination);
void ConvertIEEEToFloat(void *source, void *destination);
void ConvertIEEEToDouble(void *source, void *destination);
```

LIMITATIONS

The libascii interface code is code that we found useful and is offered as is for your use. It's intended to be used as an assistance in getting your applications running as quickly as possible.

The following are some of the known restrictions:

- Not all the C functions are supported.

  The file libascii.lst contains a list of supported ascii run time library routines.

- For some of the supported functions there are known restrictions, as follows:

  - GETOPT function

    -- The libascii getopt() function is not thread-safe. The second argument is changed for a short period of time from EBCDIC to ASCII and then back to EBCDIC.

  - PRINTF family functions

    -- The "%$n" specification is not supported in the format string.
    -- These functions are limited to 2048 bytes of output. To increase the size of the strings and output supported change #define MAXSTRING_a in global_a.h

  - SCANF family functions

    -- The "%$n" specification is not supported in the format string.
    -- The maximum number of arguments supported is 20.
    -- These functions are limited to 2048 bytes of input. To increase the size of the strings and output supported change #define MAXSTRING_a in global_a.h

- The interface layer is not NLS-enabled; it only supports ISO8859-1 <-> IBM1040-1 character set conversions.

FAQs

The following are frequently asked questions:

1. I compiled my code using -D__STRING_CODE_SET__="ISO8859-1" and included the _Ascii_a.h. The printf output to stdout is unreadable.

   Response: Not all the header files were included. For example libascii requires <stdio.h> to be included to use the libascii version of printf(). If your application uses ctime() and you wish to use the libascii ctime then <time.h> must be included.

   Another possible error is _Ascii_a.h must be included after all the other header files.

2. I compiled my code using -D__STRING_CODE_SET__="ISO8859-1" and included the _Ascii_a.h. The link edit of my application fails.

   Response: The application's "make" file needs to be modified to specify libasii.a on the link edit.

```
CHANGE HISTORY
     06/08/97 - Added gcvt() fuction
     06/08/97 - Fixed ctime already defined error message when time.h
                is included.
     06/12/97 - Added undef for putc and putchar if already defined.
     06/18/97 - Removed K&R CTYPE macros isxxx() from _Ascii_a.h.
                Too many problems with callers passing parms
                like c++ which incremented c more than once.
     06/23/97 - Improved all printf() to support strings larger than
                1000 bytes.  MAXSTRING_a in global_a.h can be set to
                the maximum printf() string the application requires.
     07/22/97 - Fixed gcvt and fcvt to return the same buffer passed
                on input. Fixed bug to printf() and sprintf() to print
                more than 199 characters.
     10/16/97 - Fixed printf family of functions to handle %c when the
                argument was '\0'.  (Example: frpintf(fp,"1 %c 2",'\0'); )
     10/30/97 - Fixed localconv() program check and inet_addr() from looping
```

# Appendix C. Special Notices

This publication is intended to help Lotus Domino application developers to migrate applications written in C to Domino for S/390. The information in this publication is not intended as the specification of any programming interfaces that are provided by Lotus Domino for S/390. See the publications delivered with the Domino for S/390 server code for the formal product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

| | |
|---|---|
| AIX | AS/400 |
| DB2 | IBM |
| Language Environment | MVS |
| OpenEdition | OS/2 |
| OS/390 | S/390 |
| System/390 | VisualAge |

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Lotus Notes C API 4.5.1 Toolkit is copyrighted by and used with permission of Lotus Development Corporation. Lotus Notes is a registered trademark and Domino is a trademark of Lotus Development Corporation.

Other company, product, and service names may be trademarks or service marks of others.

# Appendix D. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## D.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 67.

- *Porting Applications to the OpenEdition MVS Platform*, GG24-4473
- *Lotus Domino Server Release 4.5 on S/390: Installation, Customization and Administration*, SG24-2083 (available after first quarter 1998)
- *OS/390 Release 3 OpenEdition Installation and Customization*, SG24-2087 (available after first quarter 1998)

## D.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

| CD-ROM Title | Subscription Number | Collection Kit Number |
|---|---|---|
| System/390 Redbooks Collection | SBOF-7201 | SK2T-2177 |
| Networking and Systems Management Redbooks Collection | SBOF-7370 | SK2T-6022 |
| Transaction Processing and Data Management Redbook | SBOF-7240 | SK2T-8038 |
| Lotus Redbooks Collection | SBOF-6899 | SK2T-8039 |
| Tivoli Redbooks Collection | SBOF-6898 | SK2T-8044 |
| AS/400 Redbooks Collection | SBOF-7270 | SK2T-2849 |
| RS/6000 Redbooks Collection (HTML, BkMgr) | SBOF-7230 | SK2T-8040 |
| RS/6000 Redbooks Collection (PostScript) | SBOF-7205 | SK2T-8041 |
| RS/6000 Redbooks Collection (PDF Format) | SBOF-8700 | SK2T-8043 |
| Application Development Redbooks Collection | SBOF-7290 | SK2T-8037 |

## D.3 Other Publications

These publications are also relevant as further information sources:

- *OS/390 C/C++ Programming Guide*, SC09-2362
- *OS/390 C/C++ User's Guide*, SC09-2361
- *OS/390 C/C++ Language Reference*, SC09-2360
- *OS/390 C/C++ Run-Time Library Reference*, SC28-1663
- *OS/390 C/C++ Compiler and Run-Time Migration Guide*, SC09-2359
- *Debug Tool User's Guide and Reference*, SC09-2137
- *Language Environment for OS/390: Programming Guide*, SC28-1939
- *Language Environment for OS/390: Programming Reference*, SC28-1940
- *Language Environment for OS/390: Debugging Guide and Run-Time Messages*, SC28-1942
- *OS/390 OpenEdition Messages and Codes*, SC28-1908
- *OS/390 OpenEdition User's Guide*, SC28-1891

- *OS/390 OpenEdition Command Reference*, SC28-1892
- *OS/390 OpenEdition Programming Tools*, SC28-1904
- *OS/390 OpenEdition Programming: Assembler Callable Services Reference*, SC28-1899
- *OS/390 MVS System Messages, Volume 2 (ASB - EWX)*, GC28-1785
- *OS/390 MVS: System Codes*, GC28-1780
- *DFSMS/MVS Network File System User's Guide*, SC26-7028

These documents are provided as Notes databases with the Notes C API Toolkit:

- *Notes C API 4.51 User Guide (api451ug.nsf)*
- *Notes C API 4.51 Reference (api451re.nsf)*

See also the following Web sites:

IBM site for Lotus Domino for S/390
`http://www.s390.ibm.com/products/domino`

OS/390 UNIX System Services
`http://www.s390.ibm.com/oe/`

S/390 Partners in Development
`http://www.s390.ibm.com/products/s390da/`

IBM Home Page
`http://www.ibm.com/`

Lotus Home Page
`http://www.lotus.com/`

Application Partners, Inc.
`http://www.application-partners.com/`

# How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com.

## How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** —to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** —send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

**Redpieces**

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

## How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** —send orders to:

| | **IBMMAIL** | **Internet** |
|---|---|---|
| In United States: | usib6fpl at ibmmail | usib6fpl@ibmmail.com |
| In Canada: | caibmbkz at ibmmail | lmannix@vnet.ibm.com |
| Outside North America: | dkibmbsh at ibmmail | bookshop@dk.ibm.com |

- **Telephone orders**

| | |
|---|---|
| United States (toll free) | 1-800-879-2755 |
| Canada (toll free) | 1-800-IBM-4YOU |
| Outside North America | (long distance charges apply) |
| (+45) 4810-1320 - Danish | (+45) 4810-1020 - German |
| (+45) 4810-1420 - Dutch | (+45) 4810-1620 - Italian |
| (+45) 4810-1540 - English | (+45) 4810-1270 - Norwegian |
| (+45) 4810-1670 - Finnish | (+45) 4810-1120 - Spanish |
| (+45) 4810-1220 - French | (+45) 4810-1170 - Swedish |

- **Mail Orders** —send orders to:

| | | |
|---|---|---|
| IBM Publications<br>Publications Customer Support<br>P.O. Box 29570<br>Raleigh, NC 27626-0570<br>USA | IBM Publications<br>144-4th Avenue, S.W.<br>Calgary, Alberta T2P 3N5<br>Canada | IBM Direct Services<br>Sortemosevej 21<br>DK-3450 Allerød<br>Denmark |

- **Fax** —send orders to:

| | |
|---|---|
| United States (toll free) | 1-800-445-9269 |
| Canada | 1-403-267-4455 |
| Outside North America | (+45) 48 14 2207 (long distance charge) |

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** —ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to `softwareshop@vnet.ibm.com`
- **On the World Wide Web**

| | |
|---|---|
| Redbooks Web Site | http://www.redbooks.ibm.com |
| IBM Direct Publications Catalog | http://www.elink.ibmlink.ibm.com/pbl/pbl |

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank).

**Redpieces**

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.htm`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

## IBM Redbook Order Form

**Please send me the following:**

| Title | Order Number | Quantity |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

□ Invoice to customer number

□ Credit card number

Credit card expiration date Card issued to Signature

**We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.**

# Glossary

**AIX**. Advanced Interactive Executive (IBM's flavor of UNIX)

**ANSI (American National Standards Institute)**. An organization sponsored by the Computer and Business Equipment Manufacturers Association for establishing voluntary standards.

**API**. application program interface

**ASCII**. American National Standard Code for Information Interchange.

**C**. UNIX system-programming language

**DLL**. dynamic link library (OS/2)

**EBCDIC**. extended binary coded decimal interchange code

**FTP**. file transfer program

**hex**. Hexadecimal.

**hexadecimal (hex)**. Pertaining to a numbering system with base of 16; valid numbers use the digits 0 through 9 and characters A through F, where *A* represents 10 and *F* represents 15.

**HFS**. Hierarchical File System (DFSMS/MVS, POSIX 1003.1(a) compliant file system)

**IBM**. International Business Machines Corporation

**IEEE**. Institute of Electrical and Electronics Engineers

**ISPF**. interactive system productivity facility (MVS & VM)

**ITSO**. International Technical Support Organization

**LMBCS**. Lotus multibyte character set

**LOTUS**. Lotus Development Corporation (software developer)

**NFS**. network file system (USA, Sun Microsystems Inc)

**S/390**. System/390. Any ES/9000 system including its associated I/O devices and operating system(s).

**TCP/IP**. See *transmission control protocol/internet protocol.*

**transmission control protocol/internet protocol (TCP/IP)**. A public domain networking protocol with standards maintained by US Department of Defense to allow unlike vendor systems to communicate.

**TSO**. time sharing option

**UNIX**. an operating system developed at Bell Laboratories (trademark of UNIX System Laboratories, licensed exclusively by X/Open Company, Ltd.)

**URL**. Uniform Resource Locator

# Index

## Special Characters



## Numerics



## A



## B



## C



## D



## E

## F



## G



## H



## I



## K



## L



## M



## N

## U



## V



## W



## X



## Y

# ITSO Redbook Evaluation

Porting C Applications to Lotus Domino on S/390
SG24-2092-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

**Please rate your overall satisfaction** with this book using the scale:
**(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)**

**Overall Satisfaction** _____

**Please answer the following questions:**

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

**Comments/Suggestions:** **( THANK YOU FOR YOUR FEEDBACK! )**

_____

_____

_____

_____

_____

SG24-2092-00
Printed in the U.S.A.

## Artwork Definitions



## Table Definitions



## Figures

| | | | | |
|---|---|---|---|---|
| COMHDR | 2092COMP | 29 | 19 | 27, 29 |
| COMBYT | 2092COMP | 33 | 20 | 32 |
| COMSEA | 2092COMP | 33 | 21 | 33 |
| COMI | 2092COMP | 34 | 22 | 33 |
| TESRC1 | 2092TEST | 38 | 23 | 38 |
| TESRC2 | 2092TEST | 39 | 24 | 39 |
| TESRC3 | 2092TEST | 39 | 25 | 39 |
| TESRC4 | 2092TEST | 39 | 26 | 39 |
| TESRC5 | 2092TEST | 40 | 27 | 40 |
| TESSTA3 | 2092TEST | 41 | 28 | 40 |
| TESSTA4 | 2092TEST | 41 | 29 | 41 |
| TESSTA5 | 2092TEST | 41 | 30 | 41 |
| TESSTA1 | 2092TEST | 42 | 31 | 41 |
| TESMSG1 | 2092TEST | 42 | 32 | 42 |
| TESMSG2 | 2092TEST | 43 | 33 | 42 |
| TESMSG3 | 2092TEST | 43 | 34 | 43 |
| TESCOR | 2092TEST | 43 | 35 | 43 |
| TESSEP | 2092TEST | 45 | 36 | 45 |
| TESHAN1 | 2092TEST | 46 | 37 | 45 |
| TESHAN2 | 2092TEST | 46 | 38 | 46, 46 |
| TESTEX1 | 2092TEST | 48 | 39 | 48 |
| TESSEG | 2092TEST | 48 | 40 | 48 |
| TESEXC1 | 2092TEST | 49 | 41 | 49 |
| TESEXC2 | 2092TEST | 49 | 42 | 49 |

**Headings**

| id | File | Page | References |
|---|---|---|---|
| ACK | 2092ACKS | ix | The Team That Wrote This Redbook 6 |
| REDBCOM | REDB$COM | x | Comments Welcome |
| SUM | 2092SUMM | 1 | Chapter 1, Summary |
| DES | 2092DESC | 3 | Chapter 2, Porting Project Overview |
| PLA | 2092PLAN | 9 | Chapter 3, Planning the Port |
| PLAENV | 2092PLAN | 9 | 3.1, The Development Environment on OS/390 |
| PLAOMVS | 2092PLAN | 10 | 3.1.2.2, OMVS 12 |
| PLAEBC | 2092PLAN | 13 | 3.2.1, ASCII/EBCDIC Differences 13 |
| INS | 2092INST | 17 | Chapter 4, Notes C API Toolkit and libascii Installation 9 |
| INSLIB | 2092INST | 18 | 4.2, libascii Installation |
| INSLWEB | 2092INST | 19 | 4.2.1, Downloading the libascii Package |
| INSCAPI | 2092INST | 21 | 4.3, Notes C API Toolkit |
| INSCWEB | 2092INST | 21 | 4.3.1, Downloading the Notes C API Package |
| INSVER | 2092INST | 23 | 4.3.5, Verifying the Environment 23, 24 |
| INSDOC | 2092INST | 23 | 4.3.6, Copying the Notes C API Toolkit Documentation |
| INSINTR | 2092INST | 24 | 4.3.7, Building and Running the intro Sample 12 |
| COM | 2092COMP | 27 | Chapter 5, Compiling the Code |
| COMMOD | 2092COMP | 27 | 5.1, Modifying the Code 14, 21 |
| COMCOD | 2092COMP | 28 | 5.2.1, Coding Rules for Comments |
| COMHDR | 2092COMP | 28 | 5.2.3, Header File Differences 13 |
| TES | 2092TEST | 37 | Chapter 6, Testing the Code |
| NRES | 2092TEST | 38 | 6.2.1, Symptoms of No Resource Compiler 38 |
| TESSTA | 2092TEST | 40 | 6.2.4, Changes for the Status Line 38, 42 |
| TESMSG | 2092TEST | 42 | 6.3, Status Missing in WIT Startup 38 |
| TESCOR | 2092TEST | 43 | 6.4, Message Strings Corrupted 45 |
| TESSEG | 2092TEST | 48 | 6.8, Segmentation Violation When Close Domino Server 7 |
| API | 2092CAPI | 51 | Appendix A, API 4.5.1 Code Readme File 15, 21, 22 |
| LIB | 2092LIBA | 57 | Appendix B, OS/390 OpenEdition libascii Read File 14, 15, 15, 19, 19, 21, 27 |
| NOTICES | SG242092 SCRIPT | 63 | Appendix C, Special Notices ii |
| BIBL | 2092BIBL | 65 | Appendix D, Related Publications |
| REDBCDR | REDB$BIB | 65 | D.2, Redbooks on CD-ROMs |
| ORDER | REDB$ORD | 67 | How to Get ITSO Redbooks 65 |

x

**Index Entries**

**Tables**

| id | File | Page | References |
|---|---|---|---|
| INSSPAC | 2092INST | 18 | 1<br>18 |
| TESHAN | 2092TEST | 47 | 2<br>47 |

**Processing Options**

Runtime values:

| | |
|---|---|
| Document fileid | SG242092 SCRIPT |
| Document type | USERDOC |
| Document style | REDBOOK |
| Profile | EDFPRF40 |
| Service Level | 0022 |
| SCRIPT/VS Release | 4.0.0 |
| Date | 98.02.23 |
| Time | 18:46:29 |
| Device | 3820A |
| Number of Passes | 3 |
| Index | YES |
| SYSVAR D | YES |
| SYSVAR G | INLINE |
| SYSVAR X | YES |

Formatting values used:

| | |
|---|---|
| Annotation | NO |
| Cross reference listing | YES |
| Cross reference head prefix only | NO |
| Dialog | LABEL |
| Duplex | YES |
| DVCF conditions file | (none) |
| DVCF value 1 | (none) |
| DVCF value 2 | (none) |
| DVCF value 3 | (none) |
| DVCF value 4 | (none) |
| DVCF value 5 | (none) |
| DVCF value 6 | (none) |
| DVCF value 7 | (none) |
| DVCF value 8 | (none) |
| DVCF value 9 | (none) |
| Explode | NO |
| Figure list on new page | YES |
| Figure/table number separation | YES |
| Folio-by-chapter | NO |
| Head 0 body text | Part |
| Head 1 body text | Chapter |
| Head 1 appendix text | Appendix |
| Hyphenation | NO |
| Justification | NO |
| Language | ENGL |
| Keyboard | 395 |
| Layout | OFF |
| Leader dots | YES |
| Master index | (none) |
| Partial TOC (maximum level) | 4 |
| Partial TOC (new page after) | INLINE |
| Print example id's | NO |
| Print cross reference page numbers | YES |
| Process value | (none) |
| Punctuation move characters | ., |
| Read cross-reference file | (none) |
| Running heading/footing rule | NONE |
| Show index entries | NO |
| Table of Contents (maximum level) | 3 |
| Table list on new page | NO |
| Title page (draft) alignment | RIGHT |
| Write cross-reference file | (none) |

**Imbed Trace**

| | |
|---|---|
| Page 0 | 2092SU |
| Page 0 | 2092VARS |
| Page 0 | REDB$POK |
| Page i | REDB$ED1 |
| Page i | 2092EDNO |
| Page i | REDB$ED2 |
| Page ix | 2092ABST |
| Page ix | 2092ACKS |
| Page ix | REDB$COM |
| Page x | 2092IMBD |
| Page x | 2092INDX |
| Page x | 2092SUMM |
| Page 2 | 2092DESC |
| Page 7 | 2092PLAN |
| Page 16 | 2092INST |
| Page 17 | 2092TRE1 |
| Page 17 | 2092TRE2 |
| Page 25 | 2092COMP |
| Page 36 | 2092TEST |
| Page 50 | 2092CAPI |
| Page 55 | 2092LIBA |
| Page 63 | 2092SPEC |
| Page 63 | REDB$SPE |
| Page 63 | 2092TMKS |
| Page 64 | 2092BIBL |
| Page 65 | REDB$BIB |
| Page 66 | REDB$ORD |
| Page 69 | 2092GLOS |
| Page 76 | REDB$EVA |