IBM i

MSPLIB IO Toolkit

Reference

Version 2

MSPLIB-03

Second Edition (April 2004)

This edition applies to Version 1, Release 0, Modification Level 0, of the MSP IO Toolkit, and to all subsequent releases and modifications until otherwise indicated in new editions. This edition applies only to reduced instruction set computer (RISC) systems.

Cegeka welcomes your comments. You can send your comments to:

Cegeka
Noorderlaan 87
2030 Antwerpen
Belgium

You can also send your comments by facsimile, or you can send your comments electronically to Cegeka. See “How to Send Your Comments” for a description of the methods.

When you send information to Cegeka, you grant Cegeka a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About This Reference

This reference provides information about the MSP IO Toolkit as it is implemented using the ILE RPG compiler with the Operating System/400® (OS/400®) operating system.

This reference covers:

- Functions

Who Should Use This Reference

This reference is for programmers who are familiar with the RPG IV programming language.

This reference provides a detailed description of the MSP IO Toolkit. It does not provide information on how to use the ILE RPG compiler or converting RPG III programs to ILE RPG. For information on those subjects, see the *ILE RPG Programmer's Guide*, SC09-2507-03.

Before using this reference, you should

- Know how to use applicable OS/400 menus and displays or Control Language (CL) commands.
- Have a firm understanding of Integrated Language Environment® as described in detail in the *ILE Concepts*, SC41-5606-05.

How To Send Your Comments

Your feedback is important in helping to provide the most accurate and high-quality information. MSP welcomes any comments about this book or any other IBM i documentation.

- If you prefer to send comments by mail, use the following address:

  Cegeka
  Noorderlaan 87
  2030 Antwerpen
  Belgium

- If you prefer to send comments by FAX, use the following number

  +32 3 54.54.909

- If you prefer to send comments electronically, use the following e-mail address:

  Midmarket.support@cegeka.be

Be sure to include the following:

- The name of the book.
- The publication number of the book.
- The page number or topic to which your comment applies.

What's New In This Release?

The MSP IO Toolkit is now part of the MSP Library set of Licensed Programs.

Chapter 1. Planning to Install the MSPLIB IO Toolkit

This chapter provides a summary of the requirements for running the MSPLIB IO Toolkit for IBM i. It discusses the network protocols and compilers supported, the delivery media, and the various components of the product.

Hardware requirements

The MSPLIB IO Toolkit for IBM i runs on any machine that is capable of running OS/400® V5R1. The machine must have sufficient storage to meet the combined requirements of the programming prerequisites, the MSPLIB IO Toolkit for IBM i, the access methods, and the application programs, whether from MSP or other vendors.

The installation requirements depend on the components that you install. You also require archiving capacity on disk, tape, or other media.

Disk space required

For the MSPLIB IO Toolkit base code allow approximately 2 MB of storage.

Software requirements

The minimum supported software level is OS/400 Version 5 Release 1. Later levels, if any, will be supported.

Compilers supported for the MSPLIB IO Toolkit

The MSPLIB IO Toolkit for IBM i supports the following compilers, which are all part of the WebSphere Development Toolset, part number 5722WDS:

- ILE RPG IV - Product option 31

IBM i install libraries and directories

The MSPLIB IO Toolkit for IBM i installs into the OS/400 library structure. It does not use directories in the IFS root file system.

MSPLIB IO Toolkit for IBM i library structure

Installation of the MSPLIB IO Toolkit for IBM i creates the following libraries.

MSPLIB
The MSPLIB IO Toolkit base product library.

Delivery

The MSPLIB IO Toolkit is supplied as a savefile, which can be downloaded from:

http://systemi.cegeka.be/MSPLIB

MSPLIB IO Toolkit for IBM i components

The components are as follows:

Base Support to enable you to install the various MSPLIB Toolkits.

IO Toolkit (Option 3)

The MSPLIB IO Toolkit programs. The prototypes are supplied in the QPRTSRC source physical file and executables are supplied in the MSPLIB library.

Documentation

PDF versions of the books are supplied on the website http://systemi.cegeka.be/MSPLIB. They are not installable. You can read them directly from the web or you can copy them to your system. For example, you can copy them to your PC hard drive, or store them in the AS/400 IFS and then access them using a network drive.

Chapter 2. Installing the MSPLIB IO Toolkit for IBM i

This chapter describes installation of the MSPLIB IO Toolkit for IBM i. It covers the following topics:

- “Before installation”
- “Installation procedure”
- “Verifying the installation”

In addition to reading the information in this chapter, refer the latest information available on the MSPLIB Toolkit Web site at:

http://systemi.cegeka.be/MSPLIB

Before installation

This section describes how to install the MSPLIB IO Toolkit for IBM i. Use the following procedure only if you have never installed MSPLIB IO Toolkit on your system before.

To run the MSPLIB IO Toolkit for IBM i you must have the OS/400 V5R1 operating system installed on your machine.

Setting system values

Before installing the MSPLIB IO Toolkit for IBM i, use the DSPSYSVAL command to check that the following system values are set to the requirements of your enterprise:

- QCCSID
- QUTCOFFSET
- QSYSLIBL
- QALWOBJRST
- QSHRMEMCTL

You can change these values, if necessary, using the CHGSYSVAL command.

QCCSID

Every field might have a specific coded-character set identifier (CCSID). The CCSID tag identifies the code page and character set of the source. For CCSIDs that are supported on the IBM i machine, see the *IBM i National Language Support* book.

Note: The CCSID must be single-byte character set (SBCS). It must not be DBCS.

QUTCOFFSET

Check that the coordinated universal time offset (QUTCOFFSET) system value has been set, to indicate the relationship between the system time and Greenwich Mean Time (GMT). You do this using the CHGSYSVAL command.

If QUTCOFFSET is not set, it takes the default value of zero. The MSPLIB Toolkits for IBM i then assume that the local system time is universal time coordinated (UTC), that is, GMT, and time stamps accordingly.

QSYSLIBL

Ensure that QSYS2 is included in the list of libraries that make up the system part of the library list. The MSPLIB Toolkits for IBM i uses programs in this library for data conversion.

QALWOBJRST

Ensure that the QALWOBJRST system value is set to *ALL or *ALWPGMADP before you install the MSPLIB Toolkits for IBM i. If it is set to *NONE, the installation might fail.

After installation, reset QALWOBJRST to its original value to maintain system security.

Installation procedure

1. Sign on to the system with a user profile that has *ALLOBJ special authority, for example QSECOFR.

2. To install the MSPLIB IO Toolkit for IBM i issue the command:

   RSTLICPGM LICPGM(0MSPLIB) DEV(*SAVF) OPTION(3) SAVF(*library*/MSPLIB03)

   where:
   - 0MSPLIB is the product identifier for the MSPLIB IO Toolkit for IBM i
   - *library* is the location where you stored the downloaded savefile

Note:
1. You can install only one instance of the MSPLIB IO Toolkit for IBM i in each partition of your server.
2. If installation of the MSPLIB IO Toolkit fails, remove any partly-installed objects before attempting to reinstall.

Verifying the installation

To ensure that the product has loaded correctly, issue the Display Software Resources (DSPSFWRSC) command and check that the licensed program 0MSPLIB is listed. If you have installed the base and the IO Toolkit, you should see:

```
Resource
   ID      Option  Feature  Description
0MSPLIB    *BASE    5050    MSP - http://www.msp.be
0MSPLIB    3        5103    MSP - IO Toolkit
```

If you press F11 while viewing the Display Software Resources screen, you will see the library and version number of the products installed:

```
Resource                    Feature
   ID      Option  Feature   Type     Library     Release
0MSPLIB    *BASE    5050     *CODE    MSPLIB      V1R0M0
0MSPLIB    3        5103     *CODE    MSPLIB      V1R0M0
```

Post-installation tasks

When you have correctly installed the MSPLIB IO Toolkit for IBM i on your system refer to the MSP Toolkits Web site at:

http://systemi.cegeka.be/MSPLIB

for latest product information, and to install and apply all PTFs that are recommended.

Chapter 3. Deleting MSP IO Toolkit for IBM i

There are two ways of deleting the MSP IO Toolkit:

- A *standard* deletion removes the MSP IO Toolkit
- An *entire* deletion removes all MSP Toolkits

Both types of deletion require you to be signed on to the system with a user profile that has *ALLOBJ special authority, for example QSECOFR.

Standard deletion

Perform a standard deletion of the MSP IO Toolkit for IBM i product if you want to retain your other MSPLIB Toolkits, for example, because you intend to reinstall the product at a later date.

To perform this deletion:

1. Ensure that no locks are held on the library MSPLIB.
2. Use the Delete Licensed Program (DLTLICPGM) command to delete the specific option. To delete the MSPLIB IO Toolkit product, issue the command:

   DLTLICPGM LICPGM(0MSPLIB) OPTION(3)

Entire deletion

You can delete the MSPLIB Toolkits entirely.

To perform this deletion:

1. Ensure that no locks are held on the library MSPLIB.
2. Use the Delete Licensed Program (DLTLICPGM) command to delete the specific option. To delete the MSPLIB Toolkits, issue the command:

   DLTLICPGM LICPGM(0MSPLIB)

Chapter 4. Functions

IFSChangeOwner (IFS Change Owner)

boolean = IFSChangeOwner(string: string{: string})

This procedure changes the owner of an IFS file.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSChangeOwner   PR              N
 D  FileName                   255    Varying Const
 D  User                       255    Varying Const
 D  Group                      255    varying Const Options(*NoPass)

 Dvalue            S               N

/free
 value = IFSChangeOwner('/home/myfile.ext': 'QPGMR');
 // Changes the owner to QPGMR

 value = IFSChangeOwner('/home/myfile.ext'
                                        : 'MYUSER': 'USRGRP');
 // Changes the owner to MYUSER and sets USRGRP as group

/end-free
```

Figure 1. IFSChangeOwner example

IFSClose (IFS Close)

boolean = IFSClose(pointer)

This procedure closes a stream file indicated by the handle.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSClose          PR               N
 D  handle                           *

 Dhanle             S                *
 Dvalue             S                N

/free
 value = IFSClose(handle);
/end-free
```

Figure 2. IFSClose example

IFSCloseDir (IFS Close Directory)

boolean = IFSCloseDir(pointer)

This procedure closes a directory listing indicated by the handle.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSCloseDir      PR              N
 D  handle                         *

 Dhanle            S               *
 Dvalue            S               N

/free
 value = IFSCloseDir(handle);
/end-free
```

Figure 3. IFSCloseDir example

IFSCreate (IFS Create File)

pointer = IFSCreate(string{: integer})

This procedure creates a stream file and returns a handle to it.

Note:
1. If the codepage parameter is omitted, a default of 819 is taken.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSCreate        PR              *
 D  FileName                    255    Varying Const
 D  Codepage                     10I 0 Const Options(*NoPass)

 Dhandle           S               *

/free
 // Creates a file with the default codepage 819
 handle = IFSCreate('/home/mydir/myfile.ext');

 // Creates a file with a UTF-8 codepage
 handle = IFSCreate('/home/mydir/myfile2.ext': 1252);
/end-free
```

Figure 4. IFSCreate example

IFSCreated (IFS Created)

timestamp = IFSCreated(string)

This procedure returns the timestamp when a given stream file was created.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSCreated       PR              Z
 D  FileName                    255    Varying Const

 Dvalue            S             255    Varying

/free
 value = 'The file was created on '
       + %Char(IFSCreated('/home/mydir/myfile.ext');
/end-free
```

Figure 5. IFSCreated example

IFSDelete (IFS Delete)

boolean = IFSDelete(string)

This procedure deletes the stream file indicated by the string.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSDelete        PR              N
 D  FileName                    255    Varying Const

 Dvalue            S               N

/free
 value = IFSDelete('/home/mydir/myfile.ext');
 // Returns *ON for successful delete
/end-free
```

Figure 6. IFSDelete example

IFSDeleteDir (IFS Delete Directory)

boolean = IFSDeleteDir(string)

This procedure deletes the directory indicated by the string.

Note: The directory should be empty.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 Dcl-Pr IFSDelete Ind;
   FileName  VarChar(255) Const;
 End-Pr;

 Dcl-S value Ind;

 value = IFSDeleteDir('/home/mydir');
 // Returns *ON for successful delete
```

Figure 7. IFSDeleteDir example

IFSEOF (IFS End-Of-File)

boolean = IFSEOF(pointer)

This procedure indicates an End-Of-File condition for the specified handle.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSEOF           PR              N
 D  handle                         *

 Dvalue            S               N
 Dhandle           S               *

/free
 value = IFSEOF(handle);
 // Returns *ON for successful delete
/end-free
```

Figure 8. IFSEOF example

IFSExists (IFS Exists)

boolean = IFSExists(string)

This procedure checks if a given stream file exists

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSExists        PR              N
 D  FileName                    255    Varying Const

 Dvalue            S               N

/free
 value = IFSExists('/home/mydir/myfile.ext');
 // Returns *ON if the file exists
/end-free
```

Figure 9. IFSExists example

IFSExtract (IFS Extract)

string = IFSExtract(string: string)

This procedure extracts a requested part of the filename.

The first parameter is the filename.

The second parameter is the required part. The list of supported parts is *DRIVE, *PATH, *FILE, *FILENAME and *EXT.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSExtract       PR            255    Varying
 D  FileName                     255    Varying Const
 D  Part                         255    Varying Const

 Dvalue            S             255    Varying

/free
 value = IFSExtract('/home/mydir/myfile.ext': '*EXT');
 // Returns .ext
/end-free
```

Figure 10. IFSExtract example

IFSGetAttribute (IFS Get Attribute)

string = IFSGetAttribute(string: string{: string})

This procedure returns various attributes of a given file.

The first parameter contains the filename.

The second parameter contains the required attribute. The current list of supported attributes is *AUTH, *AUTL, *DTAAUT, *OBJAUT, *OWNER, *PGP and *USERS.

The third parameter contains the reference user. This parameter is required for the attributes *AUTH, *DTAAUT and *OBJAUT.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSGetAttribute  PR          4096    Varying
 D  FileName                  255     Varying Const
 D  Attribute                 255     Varying Const
 D  Reference                 255     Varying Const Options(*NoPass)

 Dvalue            S

/free
 value = IFSGetAttribute('/home/mydir/myfile.ext': '*OWNER');
 // Returns the owner of the IFS file

 value = IFSGetAttribute('/home/mydir/myfile.ext': '*USERS');
 // Returns a list of users with private authorities to the file

 value = IFSGetAttribute('/home/mydir/myfile.ext': '*DTAAUT':
                                                   'QSECOFR');
 // Returns the data authority of QSECOFR on the IFS file

/end-free
```

Figure 11. IFSGetAttribute example

IFSLastModified (IFS Last Modified)

timestamp = IFSLastModified(string)

This procedure returns the timestamp last modified of a given stream file.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSLastModified  PR              Z
 D  FileName                    255    Varying Const

 Dvalue            S            255    Varying

/free
 value = 'The file is last modified on '
       + %Char(IFSLastModified('/home/mydir/myfile.ext');
/end-free
```

Figure 12. IFSLastModified example

IFSOpen (IFS Open)

pointer = IFSOpen(string{: integer})

This procedure opens a stream file and returns a handle to it.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSOpen          PR              *
 D  FileName                    255    Varying Const
 D  Codepage                     10I 0 Const Options(*NoPass)

 Dhandle           S               *

/free
 handle = IFSOpen('/home/mydir/myfile.ext');
/end-free
```

Figure 13. IFSOpen example

IFSOpenDir (IFS Open Directory)

pointer = IFSOpenDir(string)

This procedure opens a directory listing and returns a handle to it.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSOpenDir       PR              *
 D  FileName                   255    Varying Const

 Dhandle           S               *

/free
 handle = IFSOpenDir('/home/mydir');
/end-free
```

Figure 14. IFSOpenDir example

IFSRead (IFS Read)

string = IFSRead(pointer{: string})

This procedure reads a stream file. The procedure can be called multiple times for reading successive blocks of 32K.

Note:
1. Stream files have no record structure; hence interpreting the data is up to the user.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSRead          PR          32767    Varying
 D  handle                        *
 D  Options                     255    Varying Const Options(*NoPass)

 Dvalue            S            255    Varying
 Dhandle           S               *

/free
 value = IFSRead(handle);
/end-free
```

Figure 15. IFSRead example

IFSReadDir (IFS Read Directory)

string = IFSReadDir(pointer)

This procedure reads a directory entry and returns its name.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSReadDir       PR          32767    Varying
 D  handle                        *

 Dvalue            S             255    Varying
 Dhandle           S                *

/free
 value = IFSReadDir(handle);
/end-free
```

Figure 16. IFSReadDir example

IFSRename (IFS Rename)

boolean = IFSRename(string: string)

This procedure renames a file

Note:
1. The source and destination directory should be the same.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSRename        PR              N
 D  FromFile                     255    Varying Const
 D  ToFile                       255    Varying Const

 Dvalue            S               N

/free
 value = IFSRename('/home/mydir/myfile.ext': '/home/mydir/myfile.bak');
/end-free
```

Figure 17. IFSRename example

IFSWaitLock (IFS Wait for Lock)

boolean = IFSWaitLock(string: integer{: integer})

This procedure waits for a file lock.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSWaitLock      PR              N
 D  FileName                     255    Varying Const
 D  Interval                      10I 0 Const
 D  Retries                       10I 0 Const Options(*NoPass)

 Dvalue            S               N

/free
 value = IFSWaitLock('/home/myfile.ext': 5: 3);
/end-free
```

Figure 18. IFSWaitLock example

IFSWrite (IFS Write)

boolean = IFSWrite(pointer: string)

This procedure writes to a stream file.

Note:
1. The stream file should be opened in write mode.

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7...+....

 DIFSWrite         PR              N
 D  handle                         *
 D  String                   32767    Varying Const

 Dhandle           S               *
 Dvalue            S               N

/free
 value = IFSWrite(handle: 'Hello world.');
/end-free
```

Figure 19. IFSWrite example

Index