ooRexx Documentation 5.0.0

Open Object Rexx

Unix Extensions Function Reference

ooRexx Documentation 5.0.0 Open Object Rexx
Unix Extensions Function Reference
Edition 2022.12.22

Author	W. David Ashley
Author	Rony G. Flatscher
Author	Mark Hessling
Author	Rick McGuire
Author	Lee Peedin
Author	Oliver Sims
Author	Erich Steinböck
Author	Jon Wolfers

Preface

This book describes extensions to the Open Object Rexx Interpreter that are specific to Unix-like operating system. The extensions are in a number of categories.

This book is intended for people who plan to develop applications using ooRexx and one or more of the Unix specific classes. In general no special knowledge of Unix programming is needed to use the Unix extensions. Therefore this book is applicable for users ranging in experience from the novice ooRexx programmer, to the experienced application developer.

This book is a reference rather than a tutorial. It assumes the reader has some exposure to object-oriented programming concepts and Rexx programming.

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

1.1. Typographic Conventions

Typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold` is used to highlight literal strings, class names, or inline code examples. For example:

> The **`Class`** class comparison methods return **`.true`** or **`.false`**, the result of performing the comparison operation.

> This method is exactly equivalent to **`subWord(`**_n_**`, 1)`**.

`Mono-spaced Normal` denotes method names or source code in program listings set off as separate examples.

> This method has no effect on the action of any `hasEntry`, `hasIndex`, `items`, `remove`, or `supplier` message sent to the collection.

```
-- reverse an array
a = .Array~of("one", "two", "three", "four", "five")

-- five, four, three, two, one
aReverse = .CircularQueue~new(a~size)~appendAll(a)~makeArray("lifo")
```

Proportional Italic is used for method and function variables and arguments.

> A supplier loop specifies one or two control variables, _index_, and _item_, which receive a different value on each repetition of the loop.

> Returns a string of length _length_ with _string_ centered in it and with _pad_ characters added as necessary to make up length.

1.2. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed, like mandatory initialization. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. How to Read the Syntax Diagrams

Throughout this book, syntax is described using the structure defined below.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ▶▶── symbol indicates the beginning of a statement.

  The ──··· symbol indicates that the statement syntax is continued on the next line.

  The ···── symbol indicates that a statement is continued from the previous line.

  The ──▶◀ symbol indicates the end of a statement.

- Required items appear on the horizontal line (the main path).

  ▶▶─(STATEMENT)──┤ *required_item* ├──▶◀

- Optional items appear below the main path.

  ▶▶─(STATEMENT)──────────────▶◀
                    └┤ *optional_item* ├┘
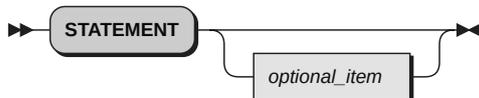
- If you can choose from two or more items, they appear vertically, in a stack. If you must choose one of the items, one item of the stack appears on the main path.

  ▶▶─(STATEMENT)──┬┤ *required_choice1* ├┬──▶◀
                    └┤ *required_choice2* ├┘
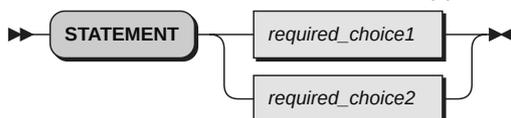
- If choosing one of the items is optional, the entire stack appears below the main path.
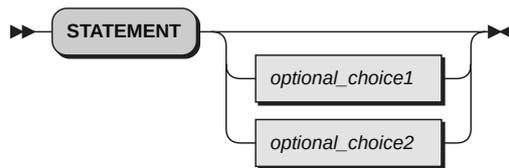
- If one of the items is the default, it is usually the topmost item of the stack of items below the main path.
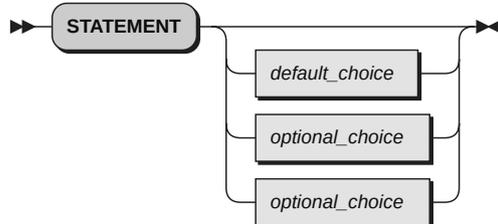
- A path returning to the left above the main line indicates an item that can be repeated.
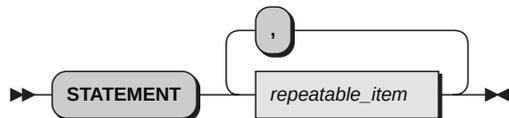
A repeat path above a stack indicates that you can repeat the items in the stack.

- A pointed rectangle around an item indicates that the item is a fragment, a part of the syntax diagram that appears in greater detail below the main diagram.

- Keywords appear in uppercase (for example, **SIGNAL**). They must be spelled exactly as shown but you can type them in upper, lower, or mixed case. Variables appear in all lowercase letters (for example, *index*). They represent user-supplied names or values.

- If punctuation marks, parentheses, arithmetic operators, or such symbols are shown, you must enter them as part of the syntax.

The following example shows how the syntax is described:
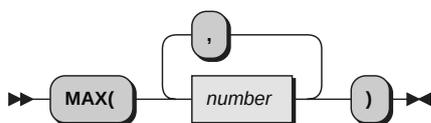
3. Getting Help and Submitting Feedback

The Open Object Rexx Project has a number of methods to obtain help and submit feedback for ooRexx and the extension packages that are part of ooRexx. These methods, in no particular order of preference, are listed below.

3.1. The Open Object Rexx SourceForge Site

Open Object Rexx utilizes SourceForge to house its source repositories, mailing lists and other project features at *https://sourceforge.net/projects/oorexx*. ooRexx uses the Developer and User mailing lists at *https://sourceforge.net/p/oorexx/mailman* for discussions concerning ooRexx. The ooRexx user is most likely to get timely replies from one of these mailing lists.

Here is a list of some of the most useful facilities provided by SourceForge.

The Developer Mailing List

Subscribe to the oorexx-devel mailing list at *https://lists.sourceforge.net/lists/listinfo/oorexx-devel* to discuss ooRexx project development activities and future interpreter enhancements. You can find its archive of past messages at *http://sourceforge.net/mailarchive/forum.php?forum_name=oorexx-devel*.

The Users Mailing List

Subscribe to the oorexx-users mailing list at *https://lists.sourceforge.net/lists/listinfo/oorexx-users* to discuss how to use ooRexx. It also supports a historical archive of past messages.

The Announcements Mailing List

Subscribe to the oorexx-announce mailing list at *https://lists.sourceforge.net/lists/listinfo/oorexx-announce* to receive announcments of significant ooRexx project events.

The Bug Mailing List

Subscribe to the oorexx-bugs mailing list at *https://lists.sourceforge.net/lists/listinfo/oorexx-bugs* to monitor changes in the ooRexx bug tracking system.

Bug Reports

You can view ooRexx bug reports at *https://sourceforge.net/p/oorexx/bugs*. To be able to create new bug reports, you will need to first register for a SourceForge userid at *https://sourceforge.net/user/registration*. When reporting a bug, please try to provide as much information as possible to help developers determine the cause of the issue. Sample program code that can reproduce your problem will make it easier to debug reported problems.

Documentation Feedback

You can submit feedback for, or report errors in, the documentation at *https://sourceforge.net/p/oorexx/documentation*. Please try to provide as much information in a documentation report as possible. In addition to listing the document and section the report concerns, direct quotes of the text will help the developers locate the text in the source code for the document. (Section numbers are generated when the document is produced and are not available in the source code itself.) Suggestions as to how to reword or fix the existing text should also be included.

Request For Enhancement

You can new suggest ooRexx features or enhancements at *https://sourceforge.net/p/oorexx/feature-requests*.

Patch Reports

If you create an enhancement patch for ooRexx please post the patch at *https://sourceforge.net/p/oorexx/patches*. Please provide as much information in the patch report as possible so that the developers can evaluate the enhancement as quickly as possible.

Please do not post bug fix patches here, instead you should open a bug report at *https://sourceforge.net/p/oorexx/bugs* and attach the patch to it.

The ooRexx Forums

The ooRexx project maintains a set of forums that anyone may contribute to or monitor. They are located at *https://sourceforge.net/p/oorexx/discussion*. There are currently three forums available: Help, Developers and Open Discussion. In addition, you can monitor the forums via email.

3.2. The Rexx Language Association Mailing List

The Rexx Language Association maintains a forum at *http://www.rexxla.org/forum.html*.

3.3. comp.lang.rexx Newsgroup

The comp.lang.rexx newsgroup at *https://groups.google.com/forum/#!forum/comp.lang.rexx* is a good place to obtain help from many individuals within the Rexx community. You can obtain help on Open Object Rexx and other Rexx interpreters and tools.

4. Related Information

See also: *Open Object Rexx: Reference*

Introduction

The RxUnixSys library provides access to some common native Unix APIs. Most of the function provided by this library should be available on most Unix/Linux systems. However, there are exceptions. For instance, the extended attribute function are not available on AIX or on any system that uses the JFS file system.

All the functions available in the RxUnixSys library require the program to contain a **::requires** directive in the source code for the program using the function(s). The statement should be coded as follows:

Example 1.1. ::Requires Directive

```
::requires "rxunixsys" LIBRARY
```

This will provide access to all the functions in the library.

Note

The examples contained in this book do not show the **::requires** directive. It is assumed that in a complete program the sample code and the directive will appear together.

Process and Thread Functions

This chapter covers the available process and thread functions.

2.1. SysGetpid

▶▶─ **SysGetpid()** ─▶◀

Gets the current process id.

This function uses the **getpid()** C API to perform the action.

Parameter:

None

Returns

Returns the current numerical process id.

> Example 2.1. SysGetpid
>
> ```
> /* get the current process id */
>
> mypid = SysGetpid()
> ```

2.2. SysGetppid

▶▶─ **SysGetppid()** ─▶◀

Gets the parent process id of the current process.

This function uses the **getppid()** C API to perform the action.

Parameter:

None

Returns

Returns the current parent process id.

> Example 2.2. SysGetppid
>
> ```
> /* get the parent process id */
>
> myppid = SysGetppid()
> ```

2.3. SysGettid

▶▶─ **SysGettid()** ─▶◀

Gets the current thread id.

This function uses the **pthread_self()** C API to perform the action.

Parameter:

None

Returns

Returns the current numerical thread id.

Example 2.3. SysGettid

```
/* get the current thread id */

mytid = SysGettid()
```

2.4. SysKill
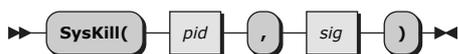
Kill the specified process using the specified signal.

This function uses the **kill()** C API to perform the action.

Parameter:

pid
   The process id to kill.

sig
   The signal to use (usually 0).

Returns

Returns the return code from the C kill() API.

Example 2.4. SysKill

```
/* kill the process */

mytid = SysKill(27313, 0)
```

User and Group Functions

This chapter covers the available user and group functions.

3.1. SysGetegid

SysGetegid()

Gets the current numerical effective group id.

This function uses the **getegid()** C API to perform the action.

Parameter:

None

Returns

Returns the current numerical effective group id.

Example 3.1. SysGetegid

```
/* get the current effective group id */

myegid = SysGetegid()
```

3.2. SysGeteuid

SysGeteuid()

Gets the current numerical effective user id.

This function uses the **geteuid()** C API to perform the action.

Parameter:

None

Returns

Returns the current numerical effective user id.

Example 3.2. SysGeteuid

```
/* get the current effective user id */

myeuid = SysGeteuid()
```

3.3. SysGetgid

SysGetgid()

Gets the current numerical group id.

This function uses the **getgid()** C API to perform the action.

Parameter:

None

Returns

Returns the current numerical group id.

> Example 3.3. SysGetgid
>
> ```
> /* get the current group id */
>
> mygid = SysGetgid()
> ```

3.4. SysGetgrgid

This function uses the **getgrgid()** C API to perform the action.

Parameter:

gid
   The numerical group id.

option
   An option specifying the information to return. The values are all case insensitive and only the first
   letter of the option value is checked. Allowed values are:
   "NAME"
      The group name.

   "GROUP"
      The numerical group id.

   "MEMBERS"
      The members of the group are returned in an array.

   "PASSWORD"
      Always returns a zero-length string.

Returns

Returns information about a group.

Returns information specified by the *option*.

> Example 3.4. SysGetgrgid
>
> ```
> /* the following examples came from a Linux laptop running Fedora */
>
> SysGetgrgid(100, "n")      -->  users
> SysGetgrgid(100, "g")      -->  100
> SysGetgrgid(100, "m")      -->  an ooRexx array with the member user names
> SysGetgrgid(100, "p")      -->  ""
> ```

3.5. SysGetgrnam

►►─ **SysGetgrnam(** ─ *grpname* ─ **,** ─ *option* ─ **)** ─►◄

Returns information about a group.

This function uses the **getgrnam()** C API to perform the action.

Parameter:

grpname
>   The group name.

option
>   An option specifying the information to return. The values are all case insensitive and only the first letter of the option value is checked. Allowed values are:
>   "NAME"
>>   The group name.
>
>   "GROUP"
>>   The numerical group id.
>
>   "MEMBERS"
>>   The members of the group are returned in an array.
>
>   "PASSWORD"
>>   Always returns a zero-length string.

Returns

Returns information specified by the *option*.

> Example 3.5. SysGetgrnam
>
> ```
> /* the following examples came from a Linux laptop running Fedora */
>
> SysGetgrnam("users", "n")      -->  users
> SysGetgrnam("users", "g")      -->  100
> SysGetgrnam("users", "m")      -->  an ooRexx array with the member user names
> SysGetgrnam("users", "p")      -->  ""
> ```

3.6. SysGetpgrp

►►─ **SysGetpgrp()** ─►◄

Gets the current numerical process group id.

This function uses the **getpgrp()** C API to perform the action.

Parameter:

None

Returns

Returns the current numerical process group id.

```
/* get the current process group id */

mypgrp = SysGetpgrp()
```

3.7. SysGetpwnam

Returns information about a user.

This function uses the **getpwnam()** C API to perform the action.

Parameter:

username
    The user name.

option
    An option specifying the information to return. The values are all case insensitive and only the first letter of the option value is checked. Allowed values are:
    "NAME"
        The user name.

    "USER"
        The numerical user id.

    "GROUP"
        The numerical group id of the user.

    "REALNAME"
        The user's full (real) name.

    "DIRECTORY"
        The user's home directory.

    "SHELL"
        The user's default shell.

    "PASSWORD"
        Always returns a zero-length string.

Returns

Returns information specified by the *option*.

Example 3.7. SysGetpwnam

```
/* the following examples came from a Linux laptop running Fedora */

SysGetpwnam("dashley", "n")      -->  dashley
SysGetpwnam("dashley", "u")      -->  500
SysGetpwnam("dashley", "g")      -->  500
SysGetpwnam("dashley", "r")      -->  David Ashley
```

```
SysGetpwnam("dashley", "d")     -->  /home/dashley
SysGetpwnam("dashley", "s")     -->  bash
SysGetpwnam("dashley", "p")     -->  ""
```

3.8. SysGetpwuid
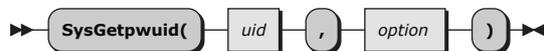
Returns information about a user.

This function uses the **getpwuid()** C API to perform the action.

Parameter:

uid

   The numerical user id.

option

   An option specifying the information to return. The values are all case insensitive and only the first letter of the option value is checked. Allowed values are:
   "NAME"

   The user name.

   "USER"

   The numerical user id.

   "GROUP"

   The numerical group id of the user.

   "REALNAME"

   The user's full (real) name.

   "DIRECTORY"

   The user's home directory.

   "SHELL"

   The user's default shell.

   "PASSWORD"

   Always returns a zero-length string.

Returns

Returns information specified by the *option*.

Example 3.8. SysGetpwuid

```
/* the following examples came from a Linux laptop running Fedora */

SysGetpwuid(500, "n")      -->  dashley
SysGetpwuid(500, "u")      -->  500
SysGetpwuid(500, "g")      -->  500
SysGetpwuid(500, "r")      -->  David Ashley
SysGetpwuid(500, "d")      -->  /home/dashley
SysGetpwuid(500, "s")      -->  bash
SysGetpwuid(500, "p")      -->  ""
```

3.9. SysGetuid

Gets the current numerical user id.

This function uses the **getuid()** C API to perform the action.

Parameter:

None

Returns

Returns the current numerical user id.

Example 3.9. SysGetuid

```
/* get the current user id */

myuid = SysGetuid()
```

3.10. SysSetegid
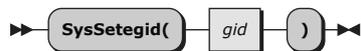
Set the numerical effective group id.

This function uses the **setegid()** C API to perform the action.

Parameter:

gid
    The new numerical effective group id.

Returns

Returns zero on success or -1 on an error.

Example 3.10. SysSetegid

```
/* set the effective group id to 520 */

call SysSetegid 520
```

3.11. SysSeteuid

Set the numerical effective user id.

This function uses the **seteuid()** C API to perform the action.

Parameter:

uid
> The new numerical effective user id.

Returns

Returns zero on success or -1 on an error.

> Example 3.11. SysSeteuid
>
> ```
> /* set the effective user id to 520 */
>
> call SysSeteuid 520
> ```

3.12. SysSetgid
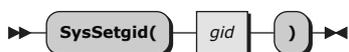
Set the numerical group id.

This function uses the **setgid()** C API to perform the action.

Parameter:

gid
> The new numerical group id.

Returns

Returns zero on success or -1 on an error.

> Example 3.12. SysSetgid
>
> ```
> /* set the group id to 520 */
>
> call SysSetgid 520
> ```

3.13. SysSetpgid

Set a numerical process id to use the specified process group id.

This function uses the **setpgid()** C API to perform the action.

Parameter:

pid
> The process id to modify.

pgid
> The new process group id.

Returns

Returns zero on success and -1 on an error.

Example 3.13. SysSetpgid

```
/* set process group id */

call SysSetpgid 9321, 520
```

3.14. SysSetpgrp

Sets the current numerical process usr id to zero and the group id to zero.

This function uses the **setpgrp()** C API to perform the action.

Parameter:

None

Returns

Returns zero on success and -1 on an error.

Example 3.14. SysSetpgrp

```
/* set the current process group id */

call SysSetpgrp
```

3.15. SysSetuid
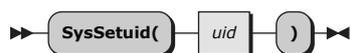
Set the numerical user id.

This function uses the **setuid()** C API to perform the action.

Parameter:

uid
    The new numerical user id.

Returns

Returns zero on success or -1 on an error.

Example 3.15. SysSetuid

```
/* set the user id to 520 */

call SysSetuid 520
```

File and File System Functions

This chapter covers the available file and file system functions.

4.1. SysAccess

Checks to see if the user has access permissions on a file.

This function uses the **access()** C API to perform the action.

Parameter:

file

    The path/file name to check access permissions.

mode

    The access permissions to check on the file. Must be a decimal number.

Returns

Returns zero on success (all access rights available) or -1 on an error.

> Example 4.1. SysGetAccess
>
> ```
> /* do others have write permission on the file? */
>
> retc = SysAccess("/home/dashley/temp.txt", 2)
> ```

4.2. SysChmod

Sets the mode bits of a file.

This function uses the **chmod()** C API to perform the action.

You must be the owner of the file or root in order for this function to be successful.

Parameter:

file

    The path/file name to to modify the mode bits.

mode

    The new and complete list of bits. Note that it is not possible to unset bits with this function except through this argument. This parameter must be a decimal number.

Returns

Returns zero on success (all access rights available) or -1 on an error.

Example 4.2. SysChmod

```
/* set the mode bits on a file */

retc = SysChmod("/home/dashley/temp.txt", "rwxrwxr-x")
retc = SysChmod("/home/dashley/temp.txt", "rwSrwsr-x")
```

4.3. SysChown

SysChown(srcpath , uid , gid)
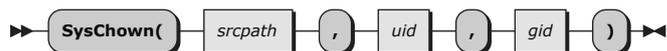
Change the user and group ownership attributes of a file.

This function uses the **chown()** C API to perform the action.

You must be the owner of the file or root in order for this function to be successful.

Parameter:

srcpath
> The path/filename of the source file to change.

uid
> The new numerical user id.

gid
> The new numerical group id.

Returns

Returns zero on success and -1 on an error.

Example 4.3. SysChown

```
/* change the ownership attributes of a file */

call SysChown "/home/username/myfile", 501, 530
```

4.4. SysChroot

SysChroot(srcpath)

Change the root directory of the current process.

This function uses the **chroot()** C API to perform the action.

Parameter:

srcpath
> The new root subdirectory base path.

Returns

Returns zero on success and -1 on an error.

```
/* change the ownership attributes of a file */

call SysChroot "/home/username/"
```

4.5. SysClose
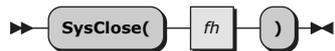
SysClose(*fh* **)**

Close the specified file handle.

This function uses the **close()** C API to perform the action.

Parameter:

fh

    The file handle to be closed.

Returns

Returns zero on success and -1 on an error.

Example 4.5. SysClose

```
/* close filehandle 0 */

call SysClose 0
```

4.6. SysEuidaccess (not available on all systems)

SysEuidaccess(*file* **,** *mode* **)**

Checks to see if the effective user has access permissions on a file.

This function uses the **euidaccess()** C API to perform the action.

Parameter:

file

    The path/file name to check access permissions.

mode

    The access permissions to check on the file. Must be a decimal number.

Returns

Returns zero on success (all access rights available) or -1 on an error.

Example 4.6. SysEuidaccess

```
/* do others have write permission on the file? */

retc = SysEuidaccess("/home/dashley/temp.txt", 2)
```

4.7. SysGetdirlist

SysGetDirList(dir)

Returns an ooRexx array of the file/directory name(s) contained in a subdirectory.

The returned array will contain ALL file and subdirectory names including files usually not displayed (hidden files/directories). The ooRexx array list of files is NOT sorted. The array will be empty if an error occurs opening the subdirectory.

This function uses the **opendir()**, **closedir()** and **readdir()** C APIs to perform the action.

Parameter:

dir
    The subdirectory to be listed.

Returns

Returns an ooRexx array.

Example 4.7. SysGetdirlist

```
/* get the subdirectory entries */

arr = SysGetdirlist("./")
do entry over arr
    say entry
    end
```

4.8. SysLchown

SysLchown(srcpath , uid , gid)
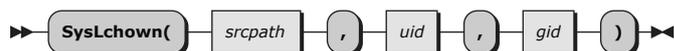
Change the user and group ownership attributes of a symbolic link.

This function uses the **lchown()** C API to perform the action.

Parameter:

srcpath
    The path/filename of the source file to change.

uid
    The new numerical user id.

gid
    The new numerical group id.

Returns

Returns zero on success and -1 on an error.

Example 4.8. SysLchown

```
/* change the ownership attributes of a symbolic link */
```

```
call SysLchown "/home/username/myfile", 501, 530
```

4.9. SysLink

```
SysLink( srcpath , targetpath )
```

Create a symbolic (hard) link.

This function uses the **link()** C API to perform the action.

Parameter:

srcpath
    The path/filename of the source file.

targetpath
    The new path/filename that will become the hard link.

Returns

Returns zero on success and -1 on an error.

Example 4.9. SysLink

```
/* create a new link */

call SysLink "/pub", "/home/username/myownpublink"
```

4.10. SysMkdirUnix

```
SysMkdirUnix( dir , mode )
```

> **Note**
>
> This function has been renamed from **SysMkdir** to **SysMkdirUnix** due to a name collision with the **rexxutil** function **SysMkDir**.

Create a subdirectory.

This function uses the **mkdir()** C API to perform the action.

Parameter:

dir
    The new subdirectory name.

mode
    The mode (decimal number) for the new subdirectory.

Returns

Returns zero on success and -1 on an error.

Example 4.10. SysMkdir

```
/* create a new subdir */

retc = SysMkdirUnix("./pub", b2d("111111101"))
```

4.11. SysRmdirUnix

> **Note**
>
> This function has been renamed from **SysRmdir** to **SysRmdirUnix** due to a name collision with the **rexxutil** function **SysRmDir**.

Remove a subdirectory.

This function uses the **rmdir()** C API to perform the action.

Parameter:

dir
    The subdirectory to remove.

Returns

Returns zero on success and -1 on an error.

Example 4.11. SysRmdir

```
/* remove a subdir */

retc = SysRmdirUnix("./pub")
```

4.12. SysStat

Returns information about a file.

This function uses the **stat64()** C API to perform the action.

Parameter:

file
    The path/file name.

option

>An option specifying the information to return. The values are all case insensitive and only the first letter of the option value is checked. Allowed values are:

>"DEVICE"

>>The file's owning device id.

>"INODE"

>>The inode number.

>"PERMISSIONS"

>>The permissions in a 10 character string similar to that produced by the shell command "ls -l".

>"NUMBERLINKS"

>>The number of links to the file.

>"UID"

>>The file's owner numerical user id.

>"GID"

>>The file's numerical group id.

>"REALDEV"

>>The file's real device id (if any).

>"SIZE"

>>The file's size in bytes.

>"ACCESS"

>>The file's last access timestamp in the form YYYY-MM-DD HH:MM:SS.

>"MODIFIED"

>>The file's last modified timestamp in the form YYYY-MM-DD HH:MM:SS.

>"CHANGED"

>>The file's last changed timestamp in the form YYYY-MM-DD HH:MM:SS.

Returns

Returns information specified by the *option*.

Example 4.12. SysStat

```
/* the following examples came from a Linux laptop running Fedora 12 */

SysStat("/home/dashley/temp.txt", "n")  -->  1
SysStat("/home/dashley/temp.txt", "u")  -->  500
SysStat("/home/dashley/temp.txt", "g")  -->  500
SysStat("/home/dashley/temp.txt", "s")  -->  427
SysStat("/home/dashley/temp.txt", "a")  -->  2009-12-02 13:22:16
```

4.13. SysSymlink
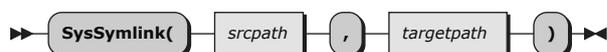
Create a symbolic (soft) link.

This function uses the **symlink()** C API to perform the action.

Parameter:

srcpath
    The path/filename of the source file.

targetpath
    The new path/filename that will become the soft link.

Returns

Returns zero on success and -1 on an error.

Example 4.13. SysSymlink

```
/* create a new symlink */

call SysSymlink "/pub", "/home/username/myownpublink"
```

4.14. SysUmask

Set the umask.

This function uses the **umask()** C API to perform the action.

Parameter:

umask
    The new umask. This MUST be a decimal number (not octal).

Returns

Returns zero on success or -1 on an error.

Example:

Example 4.14. SysUmask

```
/* set the umask */

call SysUmask 2
```

4.15. SysUname

Returns system information.

This function uses the **uname()** C API to perform the action.

Parameter:

opt

An option specifying the information to return. The values are all case insensitive and only the first letter of the option value is checked. Allowed values are:

"SYSTEM"

The system name. This is the default option if none is specified.

"NODE"

The system node (or default host) name.

"RELEASE"

The system release name.

"VERSION"

The system version.

"MACHINE"

The system machine type.

Returns

Returns information specified by the *option*.

Example 4.15. SysUname

```
/* the following examples came from a Linux laptop running Fedora 12 */

SysUname()      -->  Linux
SysUname("s")   -->  Linux
SysUname("n")   -->  bugs.ibm.com
SysUname("r")   -->  2.6.31.6-166.fc12.i686
SysUname("v")   -->  #1 SMP Wed Dec 9 11:14:59 EST 2009
SysUname("m")   -->  i686
```

4.16. SysUnLink

Removes a hard link.

This function uses the **unlink()** C API to perform the action.

Parameter:

srcpath

The path/filename of the source file to be removed.

Returns

Returns zero on success and -1 on an error.

Example 4.16. SysUnlink

```
/* remove a link */

call SysUnLink "/home/username/myownpublink"
```

4.17. SysWordexp

▶▶─(**SysWordexp(**)──(*option*)──(**)**)─◄◄

Performs a shell-like expansion of the input expression and returns the result in an array. This function will expand the characters "*", "?" and "~" (the tilde).

This function uses the **wordexp()** C API to perform the action.

Parameter:

exp

    The expression to expand.

Returns

Returns an array of the expanded results.

Example 4.17. SysWordexp

```
/* process an expansion */

arr = SysWordexp("./*.txt"
do mem over arr
    say mem
    end
```

Extended Attribute Functions

This chapter covers the available extended attribute functions. These functions are probably not available on anything but a Linux system. They are definitely not available for AIX.

5.1. SysGetxattr (not available on all systems)

►►── **SysGetxattr(** ── *fname* ── **,** ── *xname* ── **)** ──►◄

Gets an extended attribute from the specified file.

This function uses the **getxattr()** C API to perform the action.

Parameter:

fname
   The file name.

xname
   The extended attribute name.

Returns

Returns the extended attribute value on success or a zero-length string on an error.

Example 5.1. SysGetxattr

```
/* get an extended attribute */

mimetype = SysGetxattr('/home/dashley/example.txt', 'mimetype')
```

5.2. SysListxattr (not available on all systems)

►►── **SysListxattr(** ── *fname* ── **)** ──►◄

Returns an array of the extended attribute names from the specified file.

This function uses the **listxattr()** C API to perform the action.

Parameter:

fname
   The file name.

Returns

Returns an array of the extended attribute names on success. The array will be empty on an error.

Example 5.2. SysListxattr

```
/* list the extended attribute names */

names = SysListxattr('/home/dashley/example.txt')
do name over names
   say name
```

```
end
```

5.3. SysRemovexattr (not available on all systems)

Removes an extended attribute from the specified file.

This function uses the **removexattr()** C API to perform the action.

Parameter:

fname
    The file name.

xname
    The extended attribute name.

Returns

Returns 0 on success or -1 on an error.

Example 5.3. SysRemovexattr

```
/* remove an extended attribute */

retc = SysRemovexattr('/home/dashley/example.txt', 'mimetype')
```

5.4. SysSetxattr (not available on all systems)

Sets/replaces an extended attribute on the specified file.

This function uses the **removexattr()** C API to perform the action.

Parameter:

fname
    The file name.

xname
    The extended attribute name.

val
    The value to be set for the extended attribute.

Returns

Returns the 0 on success or -1 on an error.

Example 5.4. SysSetxattr

```
/* set an extended attribute */
```

```
retc = SysSetxattr('/home/dashley/example.txt', 'mimetype', 'text/plain')
```

Name Lookup Functions

This chapter covers the available name lookup functions. These functions can look up a server/service name and return information about that server/service.

6.1. SysGethostname

►► ─[**SysGethostname()**]─ ►◄

Returns the hostname of the machine.

This function uses the **gethostname()** C API to perform the action.

Parameter:

None.

Returns

Returns the hostname of the machine.

> Example 6.1. SysGethostname
>
> ```
> /* the following example comes from a Linux laptop running Fedora */
>
> SysGethostname()      -->  "wda.holmes4.com"
> ```

6.2. SysGetservbyname

►► ─[**SysGetservbyname(**]─[*name*]─(**,**)─[*proto*]─(**,**)─[*option*]─(**)**)─ ►◄

Returns information about a service.

This function uses the **getservbyname()** C API to perform the action.

Parameter:

name
    The service name (case-sensitive).

proto
    The protocol name, usually "tcp" or "udp" (case-sensitive).

option
    An option specifying the information to return. The *option* values are all case insensitive and only the first letter is significant. Allowed values are:
    "NAME"
        The service name.

    "PORT"
        The service port number.

    "ALIASES"
        The services alias names are returned in an array.

Returns

Returns information specified by the *option*.

> Example 6.2. SysGetservbyname
>
> ```
> SysGetservbyname("http", "tcp", "n")     -->  http
> SysGetservbyname("http", "tcp", "p")     -->  80
> ```

6.3. SysGetservbyport
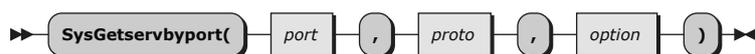
Returns information about a service.

This function uses the **getservbyport()** C API to perform the action.

Parameter:

port

    The service port number.

proto

    The protocol name, usually "tcp" or "udp" (case-sensitive).

option

    An option specifying the information to return. The *option* values are all case insensitive and only the first letter is significant. Allowed values are:

    "NAME"

        The service name.

    "PORT"

        The service port number.

    "ALIASES"

        The services alias names are returned in an array.

Returns

Returns information specified by the *option*.

> Example 6.3. SysGetservbyport
>
> ```
> SysGetservbyport(80, "tcp", "n")     -->  http
> SysGetservbyport(80, "tcp", "p")     -->  80
> ```

6.4. SysGettzname1

Returns the time zone name.

This function uses the **time.h** C header file to obtain the time zone name and is sensitive to the environment variable TZ.

Parameter:

None.

Returns

Returns the time zone name.

> Example 6.4. SysGettzname1
>
> ```
> say SysGettzname1()
> ```

6.5. SysGettzname2

 SysGettzname2() ▶◀

Returns the daylight savings time zone name.

This function uses the **time.h** C header file to obtain the time zone name and is sensitive to the environment variable TZ.

Parameter:

None.

Returns

Returns the daylight savings time zone name.

> Example 6.5. SysGettzname2
>
> ```
> say SysGettzname2()
> ```

Miscellaneous Functions

This chapter covers the available miscellaneous functions.

7.1. SysCrypt
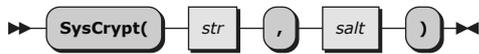
Returns an encrypted version of *str*.

This function uses the `crypt()` C API to perform the action.

Parameter:

str

    The string to be encrypted.

salt

    The encryption salt must be two characters for the default DES encryption. Depending on the platform, other salt lengths may be required for encryptions like MD5, Blowfish, SHA-256, or SHA-512.

Returns

Returns the encrypted string or a zero-length string on an error.

> Example 7.1. SysCrypt
>
> ```
> /* encrypt a password */
>
> encstr = SysCrypt("password", "3A")
> ```

7.2. SysGetSID
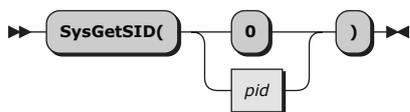
Gets the session ID of the calling process, or returns the session ID of the process with process ID *pid*.

The session ID of a process is the process group ID of the session leader.

This function uses the `getsid()` C API to perform the action.

Parameter:

pid

    `0` or a valid process ID.

Returns

Returns the session ID of the calling process if *pid* is `0`. Otherwise returns the session ID of the process with process ID *pid*.

On an error, `-1` is returned.

```
/* get the session id of the calling process */

mysid = SysGetSID(0)
```

7.3. SysGeterrno

►► ─[**SysGetErrno()**]─ ►◄

Returns the errno numeric value.

Parameter:

None

Returns

Returns the numeric errno value.

Example 7.3. SysGeterrno

```
/* get the errno */

errno = SysGeterrno()
```

7.4. SysGeterrnomsg

►► ─[**SysGetErrnoMsg(** ─ *errno* ─ **)**]─ ►◄

Returns a short description of the errno value.

Parameter:

errno
    The numeric errno value.

Returns

Returns a short description of the errno value. If the message is not available it returns the string "Unknown".

Example 7.4. SysGeterrnomsg

```
/* get the errno message */

say SysGeterrnomsg(26)
```

7.5. SysGetsizeofptr

►► ─[**SysGetsizeofptr()**]─ ►◄

Gets the size (in bits) of a pointer.

This function uses the **`sizeof(void *)`** C macro to perform the action.

Parameter:

None

Returns

Returns 32 or 64 in most cases.

Example 7.5. SysGetsizeofptr

```
/* get the pointer size */

addrmode = SysGetsizeofptr()
```

7.6. SysSetsid

►►─ **SysSetsid()** ─►◄

Set the session id.

This function uses the **`setsid()`** C API to perform the action.

Parameter:

None.

Returns

Returns the new sid or -1 on an error.

Example 7.6. SysSetsid

```
/* set the session id to zero */

call SysSetsid 0
```

7.7. SysSignal

►►─ **SysSignal(** ─ *signum* ─ **,** ─ *option* ─ **)** ─►◄

Sets the disposition of a signal to either SIG_IGN (ignore) or SIG_DFL (default).

> **Note**
>
> You should be VERY careful when calling this function as it can cause problems with the Rexx interpreter. If you don't know what you are doing then you should not call this function.

This function uses the **signal()** C API to perform the action.

Parameter:

signum
> The signal number to change. This is an integer and corresponds to the set of system signal numbers. The following is a partial list of valid values:

```
SIGHUP    1 Hangup (POSIX)
SIGINT    2 Terminal interrupt (ANSI)
SIGQUIT    3 Terminal quit (POSIX)
SIGILL    4 Illegal instruction (ANSI)
SIGTRAP    5 Trace trap (POSIX)
SIGIOT    6 IOT Trap (4.2 BSD)
SIGBUS    7 BUS error (4.2 BSD)
SIGFPE    8 Floating point exception (ANSI)
SIGKILL    9 Kill(can't be caught or ignored) (POSIX)
SIGUSR1   10 User defined signal 1 (POSIX)
SIGSEGV   11 Invalid memory segment access (ANSI)
SIGUSR2   12 User defined signal 2 (POSIX)
SIGPIPE   13 Write on a pipe with no reader, Broken pipe (POSIX)
SIGALRM   14 Alarm clock (POSIX)
SIGTERM   15 Termination (ANSI)
SIGSTKFLT 16 Stack fault
SIGCHLD   17 Child process has stopped or exited, changed (POSIX)
SIGCONT   18 Continue executing, if stopped (POSIX)
SIGSTOP   19 Stop executing(can't be caught or ignored) (POSIX)
SIGTSTP   20 Terminal stop signal (POSIX)
SIGTTIN   21 Background process trying to read, from TTY (POSIX)
SIGTTOU   22 Background process trying to write, to TTY (POSIX)
SIGURG    23 Urgent condition on socket (4.2 BSD)
SIGXCPU   24 CPU limit exceeded (4.2 BSD)
SIGXFSZ   25 File size limit exceeded (4.2 BSD)
SIGVTALRM 26 Virtual alarm clock (4.2 BSD)
SIGPROF   27 Profiling alarm clock (4.2 BSD)
SIGWINCH  28 Window size change (4.3 BSD, Sun)
SIGIO   29 I/O now possible (4.2 BSD)
SIGPWR   30 Power failure restart (System V)
```

option
> An option specifying the signal action. The values are all case insensitive and only the first letter of the option value is checked. Allowed values are:
> "DEFAULT"
>> The default action is to be set.
>
> "IGNORE"
>> Ignore this signal.

Returns

Returns previous signal value. This is meaningless to the Rexx programmer.

Example 7.7. SysSignal

```
/* Cause alarm signals to be ignored */
call SysSignal 14, "ignore"

/* Cause terminal interrupts to take the default action */
call SysSignal 2, "default"

/* Cause HUPs to be ignored */
```

```
call SysSignal 1, "I"
```

Appendix A. Notices

Any reference to a non-open source product, program, or service is not intended to state or imply that only non-open source product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Rexx Language Association (RexxLA) intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-open source product, program, or service.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-open source products was obtained from the suppliers of those products, their published announcements or other publicly available sources. RexxLA has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-RexxLA packages. Questions on the capabilities of non-RexxLA packages should be addressed to the suppliers of those products.

All statements regarding RexxLA's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

A.1. Trademarks

Open Object Rexx™ and ooRexx™ are trademarks of the Rexx Language Association.

The following terms are trademarks of the IBM Corporation in the United States, other countries, or both:

1-2-3
AIX
IBM
Lotus
OS/2
S/390
VisualAge

AMD is a trademark of Advance Micro Devices, Inc.

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the Unites States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

A.2. Source Code For This Document

The source code for this document is available under the terms of the Common Public License v1.0 which accompanies this distribution and is available in the appendix *Appendix B, Common Public License Version 1.0*. The source code is available at *https://sourceforge.net/p/oorexx/code-0/HEAD/ tree/docs/*.

The source code for this document is maintained in DocBook SGML/XML format.

The railroad diagrams were generated with the help of "Railroad Diagram Generator" located at *http:// bottlecaps.de/rr/ui*. Special thanks to Gunther Rademacher for creating and maintaining this tool.

Appendix B. Common Public License Version 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

B.1. Definitions

"Contribution" means:

1. in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and

2. in the case of each subsequent Contributor:
    a. changes to the Program, and

    b. additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

B.2. Grant of Rights

1. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

2. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

3. Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement

of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

4. Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

B.3. Requirements

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

1. it complies with the terms and conditions of this Agreement; and

2. its license agreement:

   a. effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

   b. effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

   c. states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

   d. states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

1. it must be made available under this Agreement; and

2. a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

B.4. Commercial Distribution

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified

Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

B.5. No Warranty

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

B.6. Disclaimer of Liability

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

B.7. General

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable.

However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Appendix C. Revision History

Revision 0-0   Aug 2016

    Initial creation for 5.0

Index