

rexsyslib

COLLABORATORS

	TITLE : rexsyslib		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		July 19, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	rexsyslib	1
1.1	rexsyslib.doc	1
1.2	rexsyslib.library/ClearRexxMsg	1
1.3	rexsyslib.library/CreateArgstring	2
1.4	rexsyslib.library/CreateRexxMsg	2
1.5	rexsyslib.library/DeleteArgstring	3
1.6	rexsyslib.library/DeleteRexxMsg	3
1.7	rexsyslib.library/FillRexxMsg	4
1.8	rexsyslib.library/IsRexxMsg	5
1.9	rexsyslib.library/LengthArgstring	5
1.10	rexsyslib.library/LockRexxBase	6
1.11	rexsyslib.library/UnlockRexxBase	6

Chapter 1

rexsyslib

1.1 rexsyslib.doc

```
ClearRexxMsg()  
CreateArgstring()  
CreateRexxMsg()  
DeleteArgstring()  
DeleteRexxMsg()  
FillRexxMsg()  
IsRexxMsg()  
LengthArgstring()  
LockRexxBase()  
UnlockRexxBase()
```

1.2 rexsyslib.library/ClearRexxMsg

NAME

ClearRexxMsg - Releases and clears the argument array in a RexxMsg

SYNOPSIS

```
ClearRexxMsg(msgptr, count)  
            A0      D0
```

```
VOID ClearRexxMsg(struct RexxMsg *,ULONG);
```

FUNCTION

This function will DeleteArgstring() one or more argstrings from the RexxMsg and clear the slot. The count is used to select the number of slots to clear.

INPUTS

msgptr - A pointer to a RexxMsg
count - The number of slots to be cleared. The number can be from 1 to 16. (There are 16 slots)

RESULTS

All of the slots in the given count will be cleared and the argstring will have been released.

SEE ALSO

FillRexxMsg(), DeleteRexxMsg(), DeleteArgstring(), CreateArgstring()

BUGS

1.3 rexxsyslib.library/CreateArgstring

NAME

CreateArgstring - Create an argument string structure

SYNOPSIS

```
argstr = CreateArgstring(string, length)
```

D0, A0 A0 D0

```
UBYTE *CreateArgstring(UBYTE *, ULONG);
```

FUNCTION

Allocates a REXXArg structure and copies the supplied string into it. The returned pointer points at the string part of the structure and can be treated like an ordinary string pointer. (However, care must be taken that you do not change the string)

INPUTS

string - A pointer at your input string

length - The number of bytes of your input string you wish copied.
(NOTE: You are limited to 65,535 byte strings)

RESULTS

argstr - A pointer to the argument string. The results are returned in both A0 and D0. You should always check the result as an allocation failure would cause an error.

SEE ALSO

DeleteArgstring(), LengthArgstring(), ClearRexxMsg(), FillRexxMsg()

BUGS

1.4 rexxsyslib.library/CreateRexxMsg

NAME

CreateRexxMsg - Create an ARexx message structure

SYNOPSIS

```
rexmsg = CreateRexMsg(port, extension, host)
```

D0, A0	A0	A1	D0
--------	----	----	----

```
struct RexxMsg *CreateRexxMsg(struct MsgPort *, UBYTE *, UBYTE *);
```

FUNCTION

This functions allocates an ARexx message packet. The REXXMsg consists of a standard EXEC message structure extended to include

the ARexx specific information.

INPUTS

port - A pointer to a public or private message port. This **MUST** be a valid port as this is where the message will be replied.

extension - A pointer to a NULL terminated string that is to be used as the default extension for the REXX scripts. If this is NULL, the default is "REXX"

host - A pointer to a NULL terminated string that is to be used as the default host port. The name must be the same as the name of the public message port that is to be the default host. If this field is NULL, the default is REXX.

RESULTS

rexmsg - A RexxMsg structure

NOTES

The extension and host strings must remain valid for as long as the RexxMsg exists as only the pointer to those strings are stored.

SEE ALSO

DeleteRexxMsg(), ClearRexxMsg(), FillRexxMsg()

BUGS

1.5 rexsyslib.library/DeleteArgstring

NAME

DeleteArgstring - Releases an Argstring created by CreateArgstring()

SYNOPSIS

DeleteArgstring(argstring)
A0

VOID DeleteArgstring(UBYTE *);

FUNCTION

Releases an argstring. The argstring must have been created by ARexx

INPUTS

argstring - A pointer to the string buffer of an argstring.

RESULTS

SEE ALSO

CreateArgstring(), ClearRexxMsg(), FillRexxMsg()

BUGS

1.6 rexsyslib.library/DeleteRexxMsg

NAME

DeleteRexxMsg - Releases a RexxMsg structure created by CreateRexxMsg()

SYNOPSIS

```
DeleteRexxMsg(packet)
           A0
```

```
VOID DeleteRexxMsg(struct RexxMsg *);
```

FUNCTION

The function releases an ARexx message packet that was allocated with CreateRexxMsg(). Any argument fields in the RexxMsg structure should be cleared before calling this function as it does not release them for you.

INPUTS

packet - A pointer to a RexxMsg structure allocated by CreateRexxMsg()

EXAMPLE

```
if (rmsg=CreateRexxMsg(myport,"myapp","MYAPP_PORT"))
{
    /* Do my think with rmsg */
    ClearRexxMsg(rmsg,16); /* We may not want to clear all 16 */
    DeleteRexxMsg(rmsg);
}
```

SEE ALSO

CreateRexxMsg(), ClearRexxMsg()

BUGS

1.7 rexsyslib.library/FillRexxMsg

NAME

FillRexxMsg - Fill the argument strings as needed

SYNOPSIS

```
result = FillRexxMsg(msgptr, count, mask)
D0                A0                D0                D1 [0:15]
```

```
BOOL FillRexxMsg(struct RexxMsg *,ULONG,ULONG);
```

FUNCTION

This function will convert and install up to 16 argument strings into a RexxMsg structure. The message packet's argument fields must be set to either a pointer to a NULL terminated string or an integer value. The mask, bits 0 to 15, correspond to the type of value is stored in the argument slot. If the bit is cleared, the argument is a string pointer; if the bit is set, the argument is an integer.

INPUTS

msgptr - Pointer to a RexxMsg (allocated via CreateRexxMsg)
count - The number of argument slots to fill in. This number should be from 1 to 16.

mask - A bit mask corresponding to the 16 fields that is used to determine the type of the field.

RESULTS

result - A boolean. If it is TRUE, the call worked. If it is false, some allocation did not work. All argstrings that were created will be released.

SEE ALSO

ClearRexxMsg(), CreateArgstring(), DeleteArgstring(), CreateRexxMsg()

BUGS

1.8 rexsyslib.library/IsRexxMsg

NAME

IsRexxMsg - Function to determine if a message came from ARexx

SYNOPSIS

```
result = IsRexxMsg(msgptr)
D0                      A0
```

```
BOOL IsRexxMsg(struct RexxMsg *);
```

FUNCTION

This function can be used to determine if a message came from an ARexx program.

INPUTS

msgptr - A pointer to the suspected RexxMsg.

RESULTS

result - A boolean: TRUE if it is an ARexx message, FALSE if not.

SEE ALSO

CreateRexxMsg()

BUGS

1.9 rexsyslib.library/LengthArgstring

NAME

LengthArgstring - Returns the length value stored in the argstring

SYNOPSIS

```
length = LengthArgstring(argstring)
D0                      A0
```

```
ULONG LengthArgstring(UBYTE *);
```

FUNCTION

This function returns the length value stored in the argstring. This is **NOT** the same as doing a strlen() type call on the argstring. (Note that argstrings may contain NULLs)

INPUTS

argstring - A pointer to an argstring that was created by ARexx

RESULTS

length - The length of the argstring.

EXAMPLE

SEE ALSO

CreateArgstring()

BUGS

1.10 rexsyslib.library/LockRexxBASE

NAME

LockRexxBASE - Obtain a semaphore lock on the RexxBASE structure

SYNOPSIS

```
LockRexxBASE(resource)
    DO
```

```
VOID LockRexxBASE(ULONG);
```

FUNCTION

Secures the specified resource in the ARexx library base.

INPUTS

resource - A manifest constant defining which resource to lock.
ZERO locks all resources.

NOTES

Currently, only ZERO resource type is available. You **MUST** make sure that you do not call this function with an undefined value as it may become defined at some future date and cause unwanted behavior.

SEE ALSO

UnlockRexxBASE()

BUGS

1.11 rexsyslib.library/UnlockRexxBASE

NAME

UnlockRexxBASE - Release a semaphore lock on the RexxBASE structure

SYNOPSIS

```
UnlockRexxBASE(resource)
    D0
```

```
VOID UnlockRexxBASE(ULONG);
```

FUNCTION

Releases the specified resource in the ARexx library base.

INPUTS

resource - A manifest constant defining which resource to unlock.
This value *MUST* match the value used in the matching
LockRexxBASE() call.

NOTES

Currently, only ZERO resource type is available. You *MUST* make sure that you do not call this function with an undefined value as it may become defined at some future date and cause unwanted behavior. You *MUST* make sure that you only call this function after a matching call to LockRexxBASE() was made.

SEE ALSO

LockRexxBASE()

BUGS