Rexx/EEC Reference

Version 1.4.0

Version 1.4.0

Version 1.4.0

TABLE OF CONTENTS

1. RexxEEC/Introduction [Modules]

[Top] [Modules]

DESCRIPTION

Rexx/EEC is an external function package that provides functions for Encrypting, Encoding and Compressing Rexx strings. The opposite functionality; Decrypting, Decoding and Decompressing is also provided by this function package.

USAGE

To make the external functions available add the following lines before: the first call to one of the functions in the package:

```
Call RxFuncAdd 'EECLoadFuncs', 'rexxeec', 'EECLoadFuncs'
Call EECLoadFuncs
```

TODO

- add external bzip2 compression?
- speed up BlowFish by running BlowFish_Init once for each key provided

BUGS

- there are some memory leaks

PORTABILITY

Rexx/EEC runs on Windows 9x/Me/NT/2k/XP, OS/2 3.0+ and on any Un*x platform

SEE ALSO

Rexx/EEC lives at http://rexxeec.sf.net

COPYRIGHT

2. RexxEEC/Constants [Modules]

[Top] [Modules]

NAME

Package **Constants**

DESCRIPTION

The following "constants" are defined when RexxEEC starts. By default, all constants are stored in an array with the stem preset to !REXXEEC.! This can be changed by using the 'CONSTANTPREFIX' value of EECvariable(). If you use "Procedure" on your labels, you MUST "EXPOSE !REXXEEC." or the stem you set with EECvariable() will not be visible. To reference the constants defined below, you must prefix them. So the "constant" ENCRYPT_TYPES would be, by default, referenced in your code as !REXXEEC.!ENCRYPT_TYPES.

SEE ALSO

EECvariable

ATTRIBUTES

- INTCODE - the error code from the last function call
- INTERRM - the error message from the last function call
- ENCRYPT_TYPES - a list of words identifying the valid encryption types
- COMPRESS_TYPES - a list of words identifying the valid compression types

3. Functions/Encode-Decode [Modules]

[Top] [Modules]

DESCRIPTION

The following functions encode and decode a string to and from an encoded format. Encoding generally converts 8 bit characters into printable characters for electronic transmission. As a result the encoded string is longer than the raw string.

USAGE

EECencode and EECdecode are symmetrical. ie: EECdecode(EECencode(str)) == str

NOTES

For UUencoding the length of the original string must be stored with the encoded string as the decoding mechanism has no way of knowing the original length of the string. Rexx/EEC prepends 4 bytes to the beginning of the returned encoded string which contains the original length of the string.

3.1. Encode-Decode/EECencode [Functions]

[Top] [Encode-Decode] [Functions]

NAME

EECencode

SYNOPSIS

encstr = **EECencode**(Str[, Type])

FUNCTION

Encodes Str to one of:

- a 7-bit string
- a Base64 string
- a URL string

ARGUMENTS

- Str - the Rexx string to be encoded.
- Type - Optional. The encoding mechanism to be used. 'UU' (the default) or 'BASE64' or 'URL'

RESULT

Str encoded or blank if an error.

SEE ALSO

EECdecode, !REXXEEC.!INTERRM

SOURCE

```
...
str = 'This is my string.'
encstr = eecencode( str )
Say 'UUencoded string is: encstr
```

3.2. Encode-Decode/EECdecode [Functions]

[Top] [Encode-Decode] [Functions]

NAME

EECdecode

SYNOPSIS

str = **EECdecode**(EncStr[, Type])

FUNCTION

Decodes EncStr from a UU or BASE64 string.

ARGUMENTS

- EncStr - The encoded string to decode.
- Type - Optional. The encoding mechanism to be used. 'UU' (the default) or 'BASE64'

RESULT

EncStr decoded or blank if an error.

SEE ALSO

EECencode, !REXXEEC.!INTERRM

SOURCE

```
...
str = 'This is my string.'
encstr = eecencode( str, 'BASE64' )
samestr = eecdecode( encstr, 'BASE64' )
If samestr \== str Then Say 'error encoding/decoding'
```

4. Functions/Encrypt-Decrypt [Modules]

[Top] [Modules]

DESCRIPTION

The following functions encrypt and decrypt a string using a specified algorithm and key. They provide symmetrical encryption functionality.

USAGE

Provided the same algorithm and key are used: EECdecrypt(EECencrypt(str, 'mykey', 'DES'), 'mykey', 'DES') == str

NOTES

All encryption algorithm encrypt data in chunks of various sizes. Any string not an exact multiple will have spaces appended before the encryption takes place. As a result of this behaviour it is not possible to decrypt a string to its original value without the appended spaces. To overcome this problem, the encryption mechanism in Rexx/EEC prepends 4 bytes to the beginning of the returned encrypted string which contains the original length of the string. This is used by the decryption function to return the exact length of the original string that was encrypted. The builtin DES encryption algorithm does not work on 64bit machines.

4.1. Encrypt-Decrypt/EECencrypt [Functions]

[Top] [Encrypt-Decrypt] [Functions]

NAME

EECencrypt

SYNOPSIS

encstr = **EECencrypt**(Str, Key[, Type])

FUNCTION

Encrypts a Rexx string using the specified Key and encryption Type.

ARGUMENTS

- Str - the Rexx string to be encrypted
- Key - the key to be used for encryption
- Type - Optional. DES, RIJNDAEL or BLOWFISH. BLOWFISH is default.

RESULT

The encrypted string. If an error occurs during encryption, the return value will be blank and an error available in the variable !REXXEEC.!ERRMSG.

SEE ALSO

EECdecrypt

SOURCE

```
...
encstr = eecencrypt( 'my secret data', 'mykey, 'DES' )
If encstr = '' Then Say 'Error encrypting' !REXXEEC.!INTERRM
```

4.2. Encrypt-Decrypt/EECdecrypt [Functions]

[Top] [Encrypt-Decrypt] [Functions]

NAME

EECdecrypt

SYNOPSIS

str = **EECdecrypt**(EncStr, Key[, Type])

FUNCTION

Decrypts a Rexx string using the specified Key and encryption Type.

ARGUMENTS

- EncStr - the Rexx string to be decrypted
- Key - the key to be used for decryption
- Type - Optional. DES, RIJNDAEL or BLOWFISH. BLOWFISH is default.

RESULT

The decrypted string. If an error occurs during decryption, the return value will be blank.

SEE ALSO

EECencrypt

SOURCE

```
...
str = eecdecrypt( '^^#((@#(', 'mykey' )
If str = '' Then Say 'Error decrypting' !REXXEEC.!INTERRM
```

5. Functions/Compress-Decompress [Modules]

[Top] [Modules]

DESCRIPTION

The following functions compress and decompress a string using the specified algorithm.

USAGE

Provided the same algorithm is used: EECdecompress(EECdecompress(str)) == str

NOTES

For types prefixed by "I-", the length of the original string must be stored with the compressed string as the decompression mechanism has no way of knowing the original length of the string. Rexx/EEC prepends 4 bytes to the beginning of the returned compressed string which contains the original length of the string. Types prefixed by "I-" are intended to be used by Rexx/EEC only. Strings compressed by type I-ZLIB or I-GZIP can only be decompressed by Rexx/EEC unless the 4 bytes prepending the compressed data is removed. Display the internal Rexx variable !REXXEEC.!COMPRESS_TYPES for the list of supported compression algorithms.

5.1. Compress-Decompress/EECcompress [Functions]

[Top] [Compress-Decompress] [Functions]

NAME

EECcompress

SYNOPSIS

cmpstr = **EECcompress**(Str[, Type])

FUNCTION

Compresses Str with the algorithm specified by Type.

ARGUMENTS

- Str - the Rexx string to be compressed
- Type - Optional. I-ZLIB, I-GZIP. I-ZLIB is default.

RESULT

The compressed string.

SEE ALSO

EECdecompress

SOURCE

```
...
cmpstr = eeccompress( 'my big, fat data' )
If cmpstr = '' Then Say 'Error compressing' !REXXEEC.!INTERRM
```

5.2. Compress-Decompress/EECdecompress [Functions]

[Top] [Compress-Decompress] [Functions]

NAME

EECdecompress

SYNOPSIS

str = **EECdecompress**(CmpStr[, Type])

FUNCTION

Decompresses CmpStr with the algorith specified by Type.

ARGUMENTS

- CmpStr - the Rexx string to be decompressed
- Type - Optional. I-ZLIB, I-GZIP. I-ZLIB is default.

RESULT

The decompressed string.

SEE ALSO

EECcompress

SOURCE

```
...
str = eecdecompress( '*#*@', 'ZLIB' )
If str = '' Then Say 'Error decompressing' !REXXEEC.!INTERRM
```

6. Functions/TOTP [Modules]

[Top] [Modules]

DESCRIPTION

The following functions implement Time-based One-Time Passwords

6.1. TOTP/EECTOTP [Functions]

[Top] [TOTP] [Functions]

NAME

EECtotp

SYNOPSIS

secret = EECtotp('SECRET') key = EECtotp('KEY', secret[,time_t]) valid = EECtotp('VERIFY', secret, key[,window])

FUNCTION

With 'action' equal "SECRET":

- generate a secret key

With 'action' equal "KEY":

- generate a key that is valid for 30 seconds from the optional 'time_t'

With 'action' equal "VERIFY":

- verify that the supplied 'key' is valid within???

ARGUMENTS

- time_t - Optional. the time the key will be valid as number of seconds since 1 Jan 1970. If not supplied 'now'
- secret - the secret key
- key - the key that is valid for 30 seconds
- window - the number of 30 second windows to check

RESULT

Any error will result in an empty string returned. Depending on the first argument, the 'secret', 'key' or validity

NOTES

The 'window' argument should be negative. It represents which keys are considered valid when verifying a key. 1 will only validate the key that is valid for the current 30 second window, 3 (the default), will check the current window, the previious window and the next window to come.

SOURCE

```
...
secret = eectotp( 'secret' )
Say 'Secret key is:' secret
...
key = eectotp( 'key', secret )
Say 'Time-based key is:' key
...
valid = eectotp( 'verify', secret, key )
Say 'The supplied key validity is:' valid
```

7. Functions/QRCODE [Modules]

[Top] [Modules]

DESCRIPTION

The following functions generate QRcodes

7.1. QRCODE/EECQRCODE [Functions]

[Top] [QRCODE] [Functions]

NAME

EECqrcode

SYNOPSIS

rcode = EECqrcode(string, stem[, ecc[, mask]])

FUNCTION

Generate the pixel matrix representing a QRcode for the supplied 'string' Will create the smallest sized qrcode (version) for the data supplied.

ARGUMENTS

- string - The string to be represented in the QRcode
- stem - The stem name for the Rexx array in which the pixel matrix will be stored
- ecc - Optional. Error Correction Code - a number between 0 (default) and 3 inclusive 0 - least error correction; 3 most error correction
- mask - Optional. 'AUTO' (the default) or a number between 0 and 7 inclusive Each mask pattern changes the bits according to their coordinates in the QR matrix. The purpose of a mask pattern is to make the QR code easier for a QR scanner to read.
- min - Optional. Minimum QRCode version. Must be between 1 and 40; default 1
- max - Optional. Maximum QRCode version. Must be between 1 and 40; default 40

RESULT

0 if successful, 1 if an error

NOTES

The pixel matrix is stored in a single dimension Rexx array identified by 'stem'. stem.0 contains the number of items in the array and also represents the dimension of the pixel matrix (always square). Each item in the array will be a Rexx string of stem.0 length. Each character in the item will be a 0 or 1, with 1 representing a 'black' pixel and 0 representing a 'white' pixel.

SOURCE

```
...
str = 'This is my QRCode'
rcode = eecqrcode( str, 'matrix.', ??? )
If rcode = 0 Then
   Do
      Do i = 1 To matrix.0
         Do j = 1 To matrix.0
            pixel = Substr( matrix.i, j )
            If pixel = 1 Then Call Charout ,  '#'
            Else Call Charout , ' '
         End
         Say
      End
   End
```

8. Functions/Luhn [Modules]

[Top] [Modules]

DESCRIPTION

The following functions implement the **Luhn** Mod N alogorith for check characters

9. LUHN/EECLUHN [Functions]

[Top] [Functions]

NAME

EECluhn

SYNOPSIS

checkchar = EECluhn('GENERATE', map, string) valid = EECluhn('VERIFY', map, string)

FUNCTION

With 'action' equal "GENERATE":

- generate a check character for the supplied string using the supplied map

With 'action' equal "VERIFY":

- verify that the supplied 'string' has a valid check character

ARGUMENTS

- map - a string of unique characters that can be present in the supplied string
- string - the string of characters for which a check character is to be generated or verified

RESULT

For "GENERATE" a single check character on success or empty if an error For "VERIFY", 1 indicates the string has a valid check character, 0 indicates the string has an invalid check character, -1 indicates an invalid character in string

NOTES

String can only contain characters that are present in the supplied map. Map cannot contain duplicates or space character.

SOURCE

```
...
checkchar = eecluhn( 'generate', '0123456789', '674837843' )
Say 'Check character is:' checkchar
...
valid = eecluhn( 'verify', '0123456789', '6748378434' )
Say 'The supplied string is:' valid
```

10. Functions/General [Modules]

[Top] [Modules]

DESCRIPTION

The following functions are single direction algorithms that calculate various values.

10.1. General/EECcrc [Functions]

[Top] [General] [Functions]

NAME

EECcrc

SYNOPSIS

num = **EECcrc**(InStr[, Type[,Source]])

FUNCTION

Calculates the CRC for the supplied string or file specified by Type

ARGUMENTS

- InStr - the Rexx string for which a CRC is to be calculated
- Type - Optional. See !REXXEEC.!CRC_TYPES for a full list. 16 is default.

NOTES

RESULT

The calculated CRC.

SOURCE

```
...
num = eeccrc( '*#*@', 16 )
Say 'CRC value is:' num
```

10.2. General/EECmd [Functions]

[Top] [General] [Functions]

NAME

EECmd

SYNOPSIS

num = **EECmd**(InStr, Type[, Source])

FUNCTION

Calculates the Message Digest (MD) checksum for the supplied string or file

ARGUMENTS

- InStr - the Rexx string or file for which an MD checksum is to be calculated
- Type - The Message Digest series. Currently only 5 is supported.
- Source - Optional. File or String. Default is String.

RESULT

The calculated MD.

SOURCE

```
...
str = '*#*@'
hash = eecmd( str, 5 )
Say 'MD value of string' str 'is:' hash
...
filename = 'RexxEEC.tar.gz'
hash = eecmd( filename, 5, 'File' )
Say 'MD value of file' filename 'is:' hash
```

10.3. General/EEChtdigest [Functions]

[Top] [General] [Functions]

NAME

EEChtdigest

SYNOPSIS

num = **EEChtdigest**(Action, Filename, Username, Realm[, Password])

FUNCTION

Provides similar functionality to the Apache htdigest command to manage passwords in a .htdigest file

ARGUMENTS

- Action - one of Add|Change|Delete indicating what action to carry out on the password entry
- Filename - the file to be manipulated
- Username - the user to manage
- Realm - the realm to which the user belongs
- Password - (optional) the password for the user. If not supplied the user will have to enter with keyboard

RESULT

0 on success, any other value an error

SEE ALSO

RFC2617

SOURCE

```
...
rcode = eechtdigest( 'Add', '.htdigest', 'mark', 'rexx.org' )
-- user will be prompted to enter password using keyboard
rcode = eechtdigest( 'Add', '.htdigest', 'mark', 'rexx.org', 'mypass' )
...
rcode = eechtdigest( 'C', '.htdigest', 'mark', 'rexx.org', 'newpass' )
...
```

10.4. General/EECsha [Functions]

[Top] [General] [Functions]

NAME

EECsha

SYNOPSIS

num = **EECsha**(InStr[, Type[, Source]])

FUNCTION

Calculates the SHA1 hash for the supplied string or file

ARGUMENTS

- InStr - the Rexx string or file for which a SHA1 hash is to be calculated
- Type - The SHA hash series. 1, or 256
- Source - Optional. File or String. Default is String.

RESULT

The calculated SHA1 or SHA256 hash as a hex value. To get the "real" hash value call x2c() on the result from **EECsha**.

SOURCE

```
...
str = '*#*@'
hash = eecsha( str )
Say 'SHA1 value (in hex) of string' str 'is:' hash
...
filename = 'RexxEEC.tar.gz'
hash = eecsha( filename, 1, 'File' )
Say 'SHA1 value (in hex) of file' filename 'is:' hash
```

11. Functions/PackageManagement [Modules]

[Top] [Modules]

DESCRIPTION

These functions are common to most Rexx external function packages.

11.1. PackageManagement/EECLoadFuncs [Functions]

[Top] [PackageManagement] [Functions]

NAME

EECloadfuncs

SYNOPSIS

rcode = EECloadfuncs()

FUNCTION

Loads all other RexxEEC external functions

ARGUMENTS

None

RESULT

0 in all cases

SEE ALSO

EECdropfuncs

11.2. PackageManagement/EECdropfuncs [Functions]

[Top] [PackageManagement] [Functions]

NAME

EECdropfuncs

SYNOPSIS

rcode = **EECdropfuncs**(["UNLOAD"])

FUNCTION

Cleans up RexxEEC environment and optionally will drop the external functions.

ARGUMENTS

- UNLOAD - causes the external functions to be dropped.

RESULT

0 in all cases

SEE ALSO

EECloadfuncs

11.3. PackageManagement/EECvariable [Functions]

[Top] [PackageManagement] [Functions]

NAME

EECvariable

SYNOPSIS

rcode = **EECvariable**(Variable [,NewValue])

FUNCTION

Get or set an internal RexxEEC variable.

ARGUMENTS

- Variable - name of the variable to get or set. See NOTES for
- NewValue - the new value of "Variable", if the variable is settable

RESULT

When setting a variable, then 0 if success, any other value is an error When getting a variable, then the value of the variable is returned.

NOTES

The "Variable" argument can be one of:

```
                        DEBUG (settable)
                           0 - no debugging
                           1 - all Rexx variables set by RexxEEC are displayed as they are set
                           2 - all RexxEEC functions are traced on entry with argument values and
                               on exit with the return value
                           4 - all internal RexxEEC functions are traced with their arguments
                               (really only useful for developers)
                           The values can be added together for a combination of the above details.
                        DEBUGFILE (settable)
                           Where any debugging output is written. By default this goes to
                           the system's error stream; usually 'stderr'.
                        CONSTANTPREFIX (settable)
                           The variable name prefix for all RexxEEC constants. By default this is
                           '!REXXEEC.!'. If you change this, it is useful to make the prefix result
                           in stemmed variables; this makes it far easier to EXPOSE these constants.
                        VERSION (readonly)
                           The full version details of RexxEEC in the format:
                           package version version_date
                           Where:
                              package      - the string 'rexxeec'
                              version      - package version in n.n format; eg 1.0
                              version_date - date package was released in DATE('N') format
```

SOURCE

```
...
Say 'We are running at debug level:' EECvariable( 'DEBUG' )
```

11.4. PackageManagement/EECQueryFunction [Functions]

[Top] [PackageManagement] [Functions]

NAME

EECQueryFunction

SYNOPSIS

rcode = EECqueryfunction(FunctionNamelResultArray[, Option])

FUNCTION

Populates an array of all functions supplied by this package depending on Option

ARGUMENTS

- FunctionName - the name of a function to query (no trailing period)
- ResultArray - the stem (trailing period) in which the list of functions is returned
- Option - one of 'R' (the default) for "registered" functions or 'A' for "available" functions

RESULT

0 if successful or 1 if the FunctionName is not found

NOTES

To determine if a FunctionName can be executed in your code, pass the function name as the first argument, and 'R' as the second. If the function can be called the function returns 0, otherwise it returns 1