Rexx/Vix Reference

Version 1.0

Table of Contents

Table of Contents

1. RexxVix/Introduction

[top]

DESCRIPTION

Rexx/Vix is an external function package that enables a Rexx programmer to control VMWare Server and the Virtual Machines that are running on VMWare Server.

USAGE

DERIVED FROM

Rexx/Vix was generated by RexxPackage from the Vix API C header file: vix.h

TODO

- Lots more documentation.
- Handle functions that return different properties.
- Implement callback functionality.

BUGS

Do not call VIXReleaseHandle() on the handle returned from VIXHOSTConnect() if running VMWare Server 1.0.2.

PORTABILITY

Functions that provide for callback functions will not work unless your Rexx interpreter implements the RexxCallback() API.

SEE ALSO

2. RexxVix/Constants

[top]

DESCRIPTION

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

SEE ALSO

VIXvariable

2.1. Constants/MiscellaneousFlags

[top][parent]

NAME

MiscellaneousFlags

DESCRIPTION

The following is a list of miscellaneous flags.

ATTRIBUTES

- HAVE_REXXCALLBACK - 1 if the interpreter supports RexxCallback() API
- other common constants for package

3. Functions/Host-Machines

[top]

DESCRIPTION

The following functions relate to operations on the host machine.

3.1. Host-Machines/VIXHOSTConnect

[top][parent]

NAME

VIXHOSTConnect

SYNOPSIS

jobHandle = **VIXHOSTConnect**(apiVersion, hostType, hostName, hostPort, userName, password, options, propertyListHandle, callbackProc[, clientData])

FUNCTION

This function initializes the host object. You must call this before calling any other Rexx/Vix function. The host object is used for all local Rexx/Vix operations.

ARGUMENTS

- apiVersion - The version of the API to be supported. This is defined as !REXXVIX.!VIX_API_VERSION.
- hostType - The type of host being opened. Must be !REXXVIX.!VIX_SERVICEPROVIDER_VMWARE_SERVER.
- hostName - Name of the host to which you want to connect.
- hostPort - Port number on the host to which you want to connect.
- userName - Name of user for authentication on the host.
- password - Password of user for authentication on the host.
- options - A value of zero, or see NOTES
- propertyListHandle - Must be !REXXVIX.!VIX_INVALID_HANDLE.
- callbackProc - A Rexx function that will be invoked when this function completes.
- clientData - Parameters that will be passed to the callbackProc function.

RESULT

A job handle that describes the state of this asynchronous call.

SEE ALSO

VIXHOSTDisconnect

NOTES

- This function is asynchronous. It uses a job object to report when the operation is complete. The function returns a handle to the job object immediately. When the job is signaled, the host handle is stored as the VIX_PROPERTY_JOB_RESULT_HANDLE property of the job object.
- To specify the local host (where the API client runs), pass null values for the hostName, hostPort, userName, and password parameters.
- If you are already connected to the host, a subsequent call to VixHostConnect() succeeds if you connect as the same user and use the same host name. Subsequent calls return the same handle value.
- When you initialize the host object, you can also control some Vix operations with the options parameter. The following option is supported: !REXXVIX.!VIX_HOSTOPTION_USE_EVENT_PUMP. All asynchronous event processing happens when the client calls VixPumpEvents(). The client is responsible for regularly calling VixPumpEvents(), such as in an event loop.

SOURCE

```
...
jobhandle = vixhostconnect( !REXXVIX.!VIX_API_VERSION, !REXXVIX.!VIX_SERVICEPROVIDER_VMWARE_SERVE
err = vixjobwait( jobhandle, !REXXVIX.!VIX_PROPERTY_JOB_RESULT_HANDLE, 'hosthandle' )
...
```

3.2. Host-Machines/VIXHOSTDisconnect

[top][parent]

NAME

VIXHOSTDisconnect

SYNOPSIS

Call **VIXHOSTDisconnect** hostHandle

FUNCTION

Call this function to disconnect the host. After you call this function the handle is no longer valid and you should not use it in any Rexx/Vix function. Similarly, you should not use any handles obtained from the host while it was connected.

ARGUMENTS

- hostHandle - The host handle returned by VIXJOBWait()

RESULT

None

SEE ALSO

VIXHOSTConnect

NOTES

SOURCE

```
...
jobhandle = vixhostconnect( !REXXVIX.!VIX_API_VERSION, !REXXVIX.!VIX_SERVICEPROVIDER_VMWARE_SERVE
err = vixjobwait( jobhandle, !REXXVIX.!VIX_PROPERTY_JOB_RESULT_HANDLE, 'hosthandle' )
...
Call vixhostdisconnect hosthandle
```

3.3. Host-Machines/VIXHOSTFindItems

[top][parent]

NAME

VIXHOSTFindItems

SYNOPSIS

rcode = **VIXHOSTFindItems**(hostHandle, searchType, searchCriteria, timeout, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- hostHandle -
- searchType -
- searchCriteria -
- timeout -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXHOSTFindItems( hostHandle, searchType, searchCriteria, timeout, callbackProc[, clientD
```

3.4. Host-Machines/VIXHOSTRegisterVM

[top][parent]

NAME

VIXHOSTRegisterVM

SYNOPSIS

rcode = **VIXHOSTRegisterVM**(hostHandle, vmxFilePath, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- hostHandle -
- vmxFilePath -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXHOSTRegisterVM( hostHandle, vmxFilePath, callbackProc[, clientData] )
```

3.5. Host-Machines/VIXHOSTUnregisterVM

[top][parent]

NAME

VIXHOSTUnregisterVM

SYNOPSIS

rcode = **VIXHOSTUnregisterVM**(hostHandle, vmxFilePath, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- hostHandle -
- vmxFilePath -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXHOSTUnregisterVM( hostHandle, vmxFilePath, callbackProc[, clientData] )
```

4. Functions/Job-Handles

[top]

DESCRIPTION

The following functions relate to operations on job handles.

4.1. Job-Handles/VIXJOBCheckCompletion

[top][parent]

NAME

VIXJOBCheckCompletion

SYNOPSIS

rcode = **VIXJOBCheckCompletion**(jobHandle, complete)

FUNCTION

??

ARGUMENTS

- jobHandle -
- complete -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXJOBCheckCompletion( jobHandle, complete )
```

4.2. Job-Handles/VIXJOBGetError

[top][parent]

NAME

VIXJOBGetError

SYNOPSIS

err = **VIXJOBGetError**(jobHandle)

FUNCTION

Returns the error code from the last asynchronous Rexx/Vix function call.

ARGUMENTS

- jobHandle - The handle of a job object, returned from any asynchronous Rexx/Vix function.

RESULT

The result returned by a completed asynchronous function.

SEE ALSO

VIXJOBWait, VIXJOBCheckCompletion

NOTES

The error code returned by this function is the same as the error code returned by VIXJOBWait().

SOURCE

```
...
jobhandle = vixvmpoweron( vmHandle, !REXXVIX.!VIX_VMPOWEROP_NORMAL, !REXXVIX.!VIX_INVALID_HANDLE,
Call vixjobwait jobHandle
...
err = vixjobgeterror( jobHandle )
if err \= !REXXVIX.!VIX_OK Then
   Do
      Say 'Error:' err
   End
```

4.3. Job-Handles/VIXJOBWait

[top][parent]

NAME

VIXJOBWait

SYNOPSIS

err = **VIXJOBWait**(jobHandle[, firstPropertyID, variable])

FUNCTION

This function waits until a job completes executing, optionally returning a value of a property.

ARGUMENTS

- jobHandle - A job handle returned from a call to another function.
- firstPropertyID - An optional argument specifying the property to be returned in variable.
- variable - The name of a Rexx variable that is set to the value of the property requested.

RESULT

!REXXVIX,!VIX_OK if the call succeeds, some other error value if it fails.

SOURCE

```
...
jobhandle = vixvmopen( hosthandle, vmpath, '', '' )
err = vixjobwait( jobhandle, !REXXVIX.!VIX_PROPERTY_JOB_RESULT_HANDLE, 'vmhandle' )
...
jobhandle = vixvmpoweron( vmHandle, !REXXVIX.!VIX_VMPOWEROP_NORMAL, !REXXVIX.!VIX_INVALID_HANDLE,
err = vixjobwait( jobHandle )
...
```

5. Functions/Virtual-Machines

[top]

DESCRIPTION

The following functions relate to operations on Virtual Machines.

5.1. Virtual-Machines/VIXVMCopyFileFromGuestToHost

[top][parent]

NAME

VIXVMCopyFileFromGuestToHost

SYNOPSIS

jobHandle = **VIXVMCopyFileFromGuestToHost**(vmHandle, guestPathName, hostPathName, options, propertyListHandle, callbackProc[, clientData])

FUNCTION

This function copies a file from the guest operating system to the host operating system. The virtual machine must be running while the file is copied.

ARGUMENTS

- vmHandle - Identifies a virtual machine. Call VIXVMOpen() to create a virtual machine handle.
- guestPathName - The path name of a file on a file system available to the guest.
- hostPathName - The path name of a file on a file system available to the host.
- options - Must be 0.
- propertyListHandle - Must be !REXXVIX.!VIX_INVALID_HANDLE.
- callbackProc - A Rexx function that will be invoked when this function completes.
- clientData - Parameters that will be passed to the callbackProc function.

RESULT

A job handle that describes the state of this asynchronous operation.

SEE ALSO

VIXVMCopyFileHostToGuest

NOTES

- The format of the file name depends on the guest or host operating system. For example, a path name

for a Microsoft Windows guest or host requires backslash as a directory separator, whereas a Linux guest or host requires a forward slash.
- You must call VIXVMLoginInGuest() before calling this procedure.
- The copy operation requires VMware Tools to be installed and running in the guest operating system.
- If any file fails to be copied, the operation returns an error. In this case, Rexx/Vix aborts the operation and does not attempt to copy the remaining files.

SOURCE

```
...
jobHandle = VIXVMCopyFileFromGuestToHost( vmHandle, 'c:\temp\file.dat', '/tmp/file.dat', 0, !REXX
err = vixjobwait( jobHandle )
```

5.2. Virtual-Machines/VIXVMCopyFileFromHostToGuest

[top][parent]

NAME

VIXVMCopyFileFromHostToGuest

SYNOPSIS

jobHandle = **VIXVMCopyFileFromHostToGuest**(vmHandle, hostPathName, guestPathName, options, propertyListHandle, callbackProc[, clientData])

FUNCTION

This function copies a file from the host operating system to the guest operating system. The virtual machine must be running while the file is copied.

ARGUMENTS

- vmHandle - Identifies a virtual machine. Call VIXVMOpen() to create a virtual machine handle.
- hostPathName - The path name of a file on a file system available to the host.
- guestPathName - The path name of a file on a file system available to the guest.
- options - Must be 0.
- propertyListHandle - Must be !REXXVIX.!VIX_INVALID_HANDLE.
- callbackProc - A Rexx function that will be invoked when this function completes.
- clientData - Parameters that will be passed to the callbackProc function.

RESULT

A job handle that describes the state of this asynchronous operation.

SEE ALSO

VIXVMCopyFileFromGuestToHost

NOTES

- The format of the file name depends on the guest or host operating system. For example, a path name for a Microsoft Windows guest or host requires backslash as a directory separator, whereas a Linux guest or host requires a forward slash.
- You must call VIXVMLoginInGuest() before calling this procedure.
- The copy operation requires VMware Tools to be installed and running in the guest operating system.
- If any file fails to be copied, the operation returns an error. In this case, Rexx/Vix aborts the operation and does not attempt to copy the remaining files.

SOURCE

```
...
jobHandle = VIXVMCopyFileFromHostToGuest( vmHandle, '/tmp/file.dat', 'c:\temp\file.dat', 0, !REXX
err = vixjobwait( jobHandle )
```

5.3. Virtual-Machines/VIXVMCreateSnapshot

[top][parent]

NAME

VIXVMCreateSnapshot

SYNOPSIS

rcode = **VIXVMCreateSnapshot**(vmHandle, x_name, description, options, propertyListHandle, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- x_name -
- description -
- options -
- propertyListHandle -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMCreateSnapshot( vmHandle, x_name, description, options, propertyListHandle, callback
```

5.4. Virtual-Machines/VIXVMDelete

[top][parent]

NAME

VIXVMDelete

SYNOPSIS

rcode = **VIXVMDelete**(vmHandle, deleteOptions, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- deleteOptions -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMDelete( vmHandle, deleteOptions, callbackProc[, clientData] )
```

5.5. Virtual-Machines/VIXVMGetNumRootSnapshots

[top][parent]

NAME

VIXVMGetNumRootSnapshots

SYNOPSIS

rcode = **VIXVMGetNumRootSnapshots**(vmHandle, result)

FUNCTION

??

ARGUMENTS

- vmHandle -
- result -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMGetNumRootSnapshots( vmHandle, result )
```

5.6. Virtual-Machines/VIXVMGetRootSnapshot

[top][parent]

NAME

VIXVMGetRootSnapshot

SYNOPSIS

rcode = **VIXVMGetRootSnapshot**(vmHandle, index, snapshotHandle)

FUNCTION

??

ARGUMENTS

- vmHandle -
- index -
- snapshotHandle -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMGetRootSnapshot( vmHandle, index, snapshotHandle )
```

5.7. Virtual-Machines/VIXVMInstallTools

[top][parent]

NAME

VIXVMInstallTools

SYNOPSIS

rcode = **VIXVMInstallTools**(vmHandle, options, commandLineArgs, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- options -
- commandLineArgs -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMInstallTools( vmHandle, options, commandLineArgs, callbackProc[, clientData] )
```

5.8. Virtual-Machines/VIXVMLoginInGuest

[top][parent]

NAME

VIXVMLoginInGuest

SYNOPSIS

jobHandle = **VIXVMLoginInGuest**(vmHandle, userName, password, options, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle - Identifies a virtual machine. Call VIXVMOpen() to create a virtual machine handle.
- userName - The name of a user account on the guest operating system.
- password - The password of the account identified by userName.
- options - Must be 0.
- callbackProc - A Rexx function that will be invoked when this function completes.
- clientData - Parameters that will be passed to the callbackProc function.

RESULT

A job handle that describes the state of this asynchronous call.

SEE ALSO

NOTES

- This function validates the account name and password. You must call this function before calling functions to perform operations on the guest operating system, such as VIXVMRunProgramInGuest(). If you do not call any guest functions, you do not need to call this function.
- The virtual machine must be powered on before calling this function.
- VMware Tools must be installed and running on the guest operating system before calling this function.

SOURCE

```
...
jobhandle = VIXVMLoginInGuest( vmHandle, 'mark', 'mypwd', 0, '' )
err = vixjobwait( jobhandle )
```

5.9. Virtual-Machines/VIXVMOpen

[top][parent]

NAME

VIXVMOpen

SYNOPSIS

rcode = **VIXVMOpen**(hostHandle, vmxFilePathName, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- hostHandle -
- vmxFilePathName -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMOpen( hostHandle, vmxFilePathName, callbackProc[, clientData] )
```

5.10. Virtual-Machines/VIXVMPowerOff

[top][parent]

NAME

VIXVMPowerOff

SYNOPSIS

rcode = **VIXVMPowerOff**(vmHandle, powerOffOptions, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- powerOffOptions -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMPowerOff( vmHandle, powerOffOptions, callbackProc[, clientData] )
```

5.11. Virtual-Machines/VIXVMPowerOn

[top][parent]

NAME

VIXVMPowerOn

SYNOPSIS

rcode = **VIXVMPowerOn**(vmHandle, powerOnOptions, propertyListHandle, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- powerOnOptions -
- propertyListHandle -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMPowerOn( vmHandle, powerOnOptions, propertyListHandle, callbackProc[, clientData] )
```

5.12. Virtual-Machines/VIXVMRemoveSnapshot

[top][parent]

NAME

VIXVMRemoveSnapshot

SYNOPSIS

rcode = **VIXVMRemoveSnapshot**(vmHandle, snapshotHandle, options, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- snapshotHandle -
- options -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMRemoveSnapshot( vmHandle, snapshotHandle, options, callbackProc[, clientData] )
```

5.13. Virtual-Machines/VIXVMReset

[top][parent]

NAME

VIXVMReset

SYNOPSIS

rcode = **VIXVMReset**(vmHandle, powerOnOptions, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- powerOnOptions -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMReset( vmHandle, powerOnOptions, callbackProc[, clientData] )
```

5.14. Virtual-Machines/VIXVMRevertToSnapshot

[top][parent]

NAME

VIXVMRevertToSnapshot

SYNOPSIS

rcode = **VIXVMRevertToSnapshot**(vmHandle, snapshotHandle, options, propertyListHandle, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- snapshotHandle -
- options -
- propertyListHandle -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMRevertToSnapshot( vmHandle, snapshotHandle, options, propertyListHandle, callbackPro
```

5.15. Virtual-Machines/VIXVMRunProgramInGuest

[top][parent]

NAME

VIXVMRunProgramInGuest

SYNOPSIS

rcode = **VIXVMRunProgramInGuest**(vmHandle, guestProgramName, commandLineArgs, options, propertyListHandle, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- guestProgramName -
- commandLineArgs -
- options -
- propertyListHandle -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMRunProgramInGuest( vmHandle, guestProgramName, commandLineArgs, options, propertyLis
```

5.16. Virtual-Machines/VIXVMSuspend

[top][parent]

NAME

VIXVMSuspend

SYNOPSIS

rcode = **VIXVMSuspend**(vmHandle, powerOffOptions, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- powerOffOptions -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMSuspend( vmHandle, powerOffOptions, callbackProc[, clientData] )
```

5.17. Virtual-Machines/VIXVMUpgradeVirtualHardware

[top][parent]

NAME

VIXVMUpgradeVirtualHardware

SYNOPSIS

rcode = **VIXVMUpgradeVirtualHardware**(vmHandle, options, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- options -
- callbackProc -
- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMUpgradeVirtualHardware( vmHandle, options, callbackProc[, clientData] )
```

5.18. Virtual-Machines/VIXVMWaitForToolsInGuest

[top][parent]

NAME

VIXVMWaitForToolsInGuest

SYNOPSIS

rcode = **VIXVMWaitForToolsInGuest**(vmHandle, timeoutInSeconds, callbackProc[, clientData])

FUNCTION

??

ARGUMENTS

- vmHandle -
- timeoutInSeconds -
- callbackProc -

- clientData -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXVMWaitForToolsInGuest( vmHandle, timeoutInSeconds, callbackProc[, clientData] )
```

6. Functions/Utility

[top]

DESCRIPTION

The following functions are utility functions.

6.1. Utility/VIXFreeBuffer

[top][parent]

NAME

VIXFreeBuffer

SYNOPSIS

Call **VIXFreeBuffer** p

FUNCTION

??

ARGUMENTS

- p -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
Call VIXFreeBuffer p
```

6.2. Utility/VIXGetErrorText

[top][parent]

NAME

VIXGetErrorText

SYNOPSIS

rcode = **VIXGetErrorText**(err, locale)

FUNCTION

??

ARGUMENTS

- err -
- locale -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXGetErrorText( err, locale )
```

6.3. Utility/VIXGetHandleType

[top][parent]

NAME

VIXGetHandleType

SYNOPSIS

rcode = **VIXGetHandleType**(handle)

FUNCTION

??

ARGUMENTS

- handle -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXGetHandleType( handle )
```

6.4. Utility/VIXGetProperties

[top][parent]

NAME

VIXGetProperties

SYNOPSIS

rcode = **VIXGetProperties**(handle, firstPropertyID, x_stem)

FUNCTION

??

ARGUMENTS

- handle -
- firstPropertyID -
- x_stem -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXGetProperties( handle, firstPropertyID, x_stem )
```

6.5. Utility/VIXGetPropertyType

[top][parent]

NAME

VIXGetPropertyType

SYNOPSIS

rcode = **VIXGetPropertyType**(handle, propertyID, propertyType)

FUNCTION

??

ARGUMENTS

- handle -
- propertyID -
- propertyType -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
rcode = VIXGetPropertyType( handle, propertyID, propertyType )
```

6.6. Utility/VIXPumpEvents

[top][parent]

NAME

VIXPumpEvents

SYNOPSIS

Call **VIXPumpEvents** hostHandle, options

FUNCTION

??

ARGUMENTS

- hostHandle -
- options -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
Call VIXPumpEvents hostHandle, options
```

6.7. Utility/VIXReleaseHandle

[top][parent]

NAME

VIXReleaseHandle

SYNOPSIS

Call **VIXReleaseHandle** handle

FUNCTION

??

ARGUMENTS

- handle -

RESULT

??

SEE ALSO

??

NOTES

SOURCE

```
...
Call VIXReleaseHandle handle
```

7. Functions/PackageManagement

[top]

DESCRIPTION

These functions are common to most Rexx external function packages.

7.1. PackageManagement/VIXLoadFuncs

[top][parent]

NAME

VIXLoadFuncs

SYNOPSIS

rcode = **VIXLoadFuncs**()

FUNCTION

Loads all other RexxVix external functions

ARGUMENTS

None

RESULT

0 in all cases

SEE ALSO

VIXDropFuncs

NOTES

7.2. PackageManagement/VIXDropFuncs

[top][parent]

NAME

VIXDropFuncs

SYNOPSIS

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

FUNCTION

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

ARGUMENTS

- UNLOAD - causes the external functions to be dropped.

RESULT

0 in all cases

SEE ALSO

VIXLoadFuncs

NOTES

7.3. PackageManagement/VIXVariable

[top][parent]

NAME

VIXVariable

SYNOPSIS

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

FUNCTION

Get or set an internal RexxVix 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 RexxVix are displayed as they are set
   2 - all RexxVix functions are traced on entry with argument values and
       on exit with the return value
   4 - all internal RexxVix 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 RexxVix constants. By default this is
   '!REXXVIX.!'. 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 RexxVix in the format:
   package version version_date
   Where:
      package      - the string 'rexxvix'
      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:' VIXVariable( 'DEBUG' )
```

7.4. PackageManagement/VIXQueryFunction

[top][parent]

NAME

VIXQueryFunction

SYNOPSIS

rcode = **VIXQueryFunction**(FunctionName|ResultArray[, 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