Table of Contents

□NFE System Function in APL64

Overview

□NFE supports:

- Reading and writing of text to files using encoding
- Getting and Setting file times
- Checking for file or folder existence
- Copying, Moving and Replacing of files
- Comparing two file contents using a SHA-512 hash

A character encoding is a 1 to 1 mapping of abstract glyphs (characters) to values that represent those glyphs. The values resulting from the encoding of glyphs can be persisted and transmitted without ambiguity.

References in this documentation are made to an NFE file. An NFE file is an APL64 'native' file whose content is treated as text content encoded according to a user-selected encoding option. The file extension of such a file need not be .txt, but using that extension is convenient for the association between file extension and a Windows application.

An NFE file can be read and written using:

- ⎕NFE in APL64 and APL+Win
- .Net programming languages such as C#
- Microsoft Notepad, Microsoft Word, Notepad++, etc.

APL64 system functions such as ⎕NAPPEND, ⎕FAPPEND, ⎕CFAPPEND, etc. are designed to be compatible with APL+Win and do not use standard .Net encoding.

Syntax

Syntax: [Result] ← [Larg] ⎕NFE Action [Rarg1] […]
Larg may be required depending on the Action argument
Action is a non-case-sensitive, character vector or scalar string indicating the ⎕NFE operation.
Right arguments may be required depending on the Action argument.

Exceptions

File operations can be performed only if the user account which attempts to execute the statements has the appropriate file permissions or an exception will be thrown.

Filename Requirements

The NFE methods which require a filePath argument will accept an absolute (full) or relative file (implied) name. Examples:

- 'c:\test\myFile.ext' is an absolute file name
- 'myFile.ext' is a relative file name which depends on the current folder

NFE Object Properties

Encoding

The encoding property determines how characters are encoded in a text file. The default encoding for the NFE object is UTF8. The encoding property affects the Append, Read and Write methods.

Syntax: [Result] ← ⎕NFE 'encoding' encodingType
Result: Previous Encoding Type
Action: Set the encodingType
encodingType is a non-case-sensitive character vector or scalar string:

- ASCII
- BigEndianUnicode
- Unicode
- UTF7
- UTF8
- UTF32

UTF7 encoding has been deprecated by Microsoft for security reasons.

It is the responsibility of the APL64 programmer to provide the NFE Append and Write methods with text which can be encoded according to the programmer-selected NFE encoding setting.

The UTFx encodings are Unicode encodings. The NFE encoding property option 'Unicode' is the Windows-standard UTF16 Unicode encoding.

Mixing encodings in the same text file can result in data which can only be accessed as bytes.
APL glyphs may be considered as encoded using the indices of ⎕AV, but that APL glyph mapping is not 1:1 or consistent in all APL implementations.

Example:
⎕NFE 'encoding' 'Unicode'

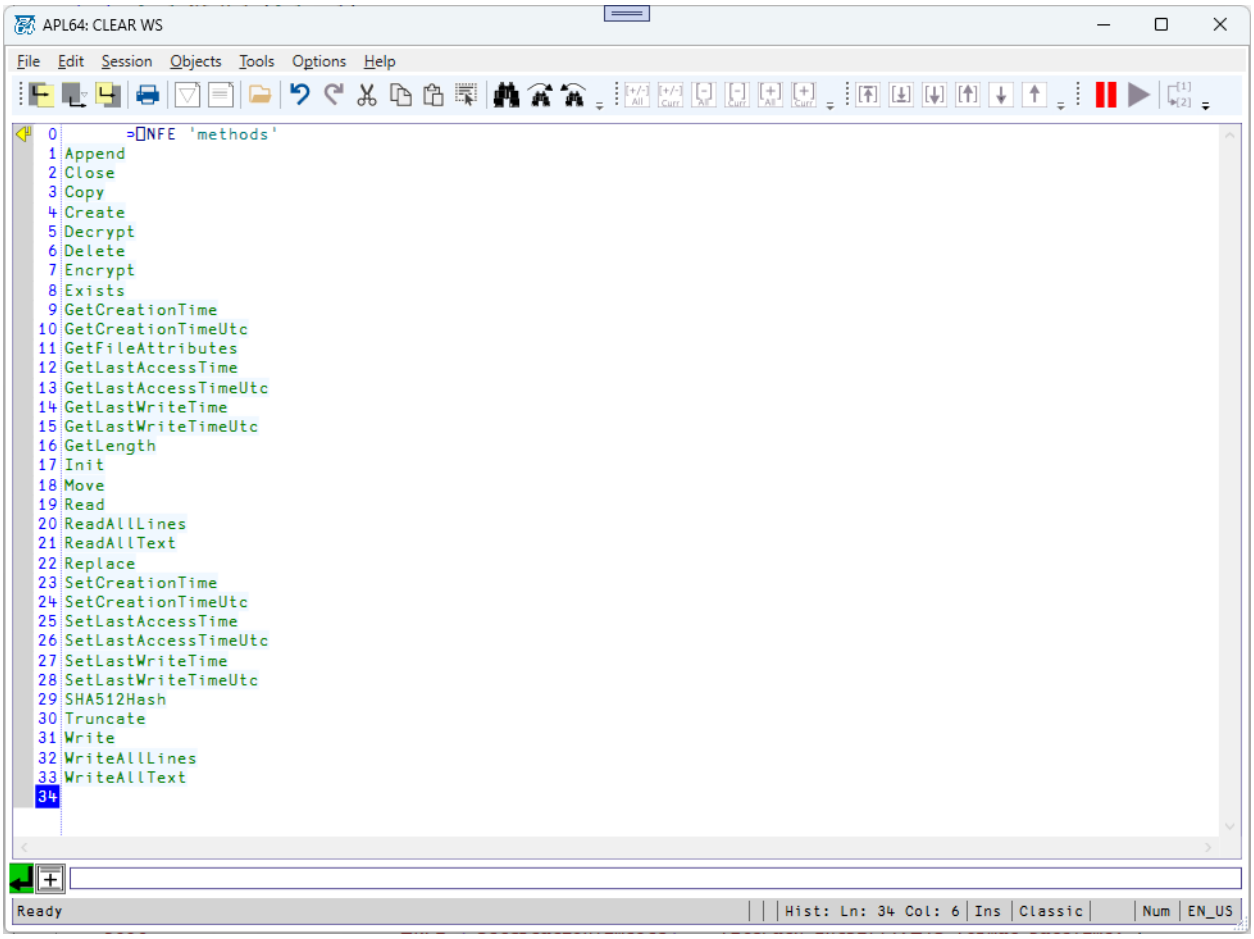
IsReadOnly

Syntax: ⎕NFE 'IsReadOnly' filePath
Action: Returns a Boolean scalar indicating if the file permission is read only, or could not be found.

Example:
⎕NFE 'IsReadOnly' 'c:\myFile.ext'

Methods

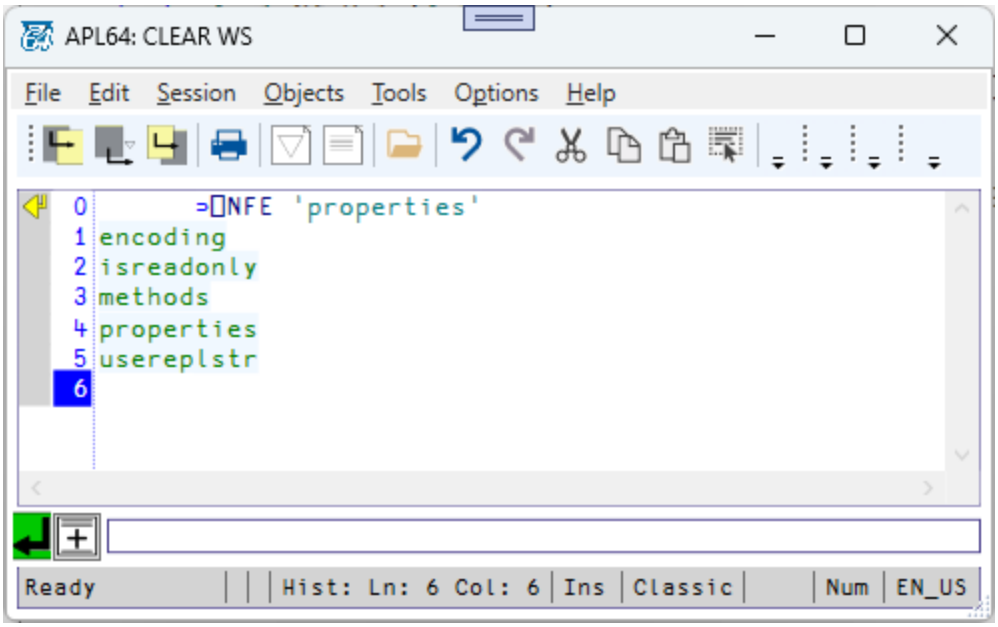
Syntax: ⎕NFE 'Methods'
The methods property returns a list of methods supported by the NFE object:



Properties

Syntax: ⎕NFE 'Properties'

The properties property returns a vector of text vectors containing the names of the NFE Object properties.



UseReplStr

The Boolean usereplstr property controls when ⎕tcnl is substituted with an operating system-dependent new line character in when APL-originated text is written to a file and vice-versa. By default, the value of this property is set to 1 to perform the character substitution.

Example:
⎕NFE 'USEREPLSTR' 0

NFE Object Methods

Append

Syntax: Text ⎕NFE 'Append' filePath
Action: Appends the encoding of text to the file specified by filePath
The targe file must exist.
Text: A character vector or scalar string

The NFE encoding property affects this method
Result: None

Examples:
⎕NFE 'encoding' 'ASCII'
'abcde' ⎕NFE 'Append' 'c:\myfile.txt'

The 'abcde' text will be encoded as ASCII and appended to the specified file. This example illustrates writing text which has Ascii code points.

⎕NFE 'encoding' 'Unicode'

' ΔΔ⍋⍒©' ⎕NFE 'Append' 'c:\myUnicodeFile.txt'

The ' ΔΔ⍋⍒©' text will be encoded as Unicode and appended to the specified file. This example illustrates writing text with Unicode code points.

⎕NFE 'encoding' 'ASCII'
'⍋ρ' ⎕NFE 'Append' 'c:\myFile.txt'

The text '⍋ρ' cannot be encoded as ASCII, so a replacement text will be appended to the file and reading this file would not permit the recovery of this text.

Copy

Syntax: srcFile ⎕NFE 'Copy' tgtFile boolOverWrite
Action:
Copies the file specified by ⎕NFE to the file specified by tgtFile
If tgtFile does not exist, this method creates the file
If tgtFile exists, it is overwritten only if boolOverwrite is 1 (True)
Result: None

Example:
'c:\myfile1.ext' ⎕NFE 'Copy' 'e:\myfilecopy.ext' 1

Close

This is an APL+Win ⎕NFE action which is deprecated in APL64.

Create

Syntax: filePath ⎕NFE 'Create' fileAccess fileShare
Result: None
Action:
Create the file specified by filePath with the fileAccessand fileSharepermissions
An exception will be thrown if the file already exists
fileAccess and fileShare are non-case-sensitive character vectors or scalar strings:

- ReadWrite
- Read
- Write

Example:
'c:\myNewFile.ext' ⎕NFE 'Create' 'ReadWrite' 'Read'

Decrypt

Syntax: ⎕NFE 'Decrypt' filePath
Result: None
Action:

Decrypts the file specified by filePath that was encrypted by the current user account using the ⎕NFE 'Encrypt' method. If the file specified by filePathwas not encrypted, this method has no effect

Example:
⎕NFE 'Decrypt' 'c:\myencryptedfile.ext'
The Decrypt action is available only when targeting the Windows operating system.

Delete

Syntax: ⎕NFE 'Delete' targetPath [isFolder] [recurseFolders]
isFolder is a Boolean scalar indicating if the targetPath is a folder [1/True]. The default value is 0/False, indicating that filePath is a file.
recurseFolders is a Boolean scalar indicating if the folder specified by targetPath and all its sub-folders should be deleted. The default value is 0/False.
Result: None
Action:
Deletes the file specified by filePath

Examples:
⎕NFE 'Delete' 'c:\myfile.ext'
Same as ⎕NFE 'Delete' 'c:\myfile.ext' 0

⎕NFE 'Delete' 'c:\myfile.ext' 0

⎕NFE 'Delete' 'c:\myfile.ext' 0 0
The fourth element of the right argument is ignored.

⎕NFE 'Delete' 'c:\myfile.ext' 0 1
The fourth element of the right argument is ignored

⎕NFE 'Delete' 'c:\topFolder\nextFolder’ 1 0
An exception will occur if nextfolder has sub-folders

⎕NFE 'Delete' 'c:\topFolder\nextFolder’ 1 1
nextFolder and any sub-folders will be deleted

Encrypt

Syntax: ⎕NFE 'Encrypt' filePath
Result: None
Action:
Encrypts the file specified by the filePath so that only the current user account can decrypt it.

Example:
⎕NFE 'Encrypt' 'c:\myfile.ext'
The Encrypt action is available only when targeting the Windows operating system.

Exists

Syntax: ⎕NFE ‘EXISTS’ filePath [isFolder]
filePath: A file name or folder name
isFolder: A Boolean scalar indicating if the filePath is a folder (1/true) or file (0/false). The default value is 0.
Action: Returns a Boolean result (0/False 1/True) indicating:

- The current user account has read permission for the filePath
- The filePath exists

Examples:
⎕NFE ‘EXISTS’ ‘c:\myFolder’ 1
⎕NFE ‘EXISTS’ ‘c:\myFolder\myFile.txt’ 0

GetCreationTime

Syntax: ⎕NFE 'GetCreationTime' filePath
Action: Returns the ⎕TS-format time of creation of the file specified by filePath

Example:
⎕NFE 'GetCreationTime' 'c:\myFile.ext'

GetCreationTimeUtc

Syntax: ⎕NFE 'GetCreationTimeUtc' filePath
Action: Returns the ⎕TS-format UTC time of creation of the file specified by filePath
Example:
⎕NFE 'GetCreationTimeUTC' 'c:\myFile.ext'

GetFileAttributes

Syntax: ⎕NFE 'GetFileAttributes' filePath
Action: Returns an integer representing the .Net FileAttributes of the file specified by filePath

Example:
⎕NFE 'GetFileAttributes' 'c:\myFile.ext'
For additional documentation refer to: FileAttributes Enum (System.IO) | Microsoft Docs

GetLastAccessTime

Syntax: ⎕NFE 'GetLastAccessTime' filePath
Action: Returns the ⎕TS-format time of the last access of the file specified by filePath

Example:
⎕NFE 'GetLastAccessTime' 'c:\myFile.ext'

GetLastAccessTimeUtc

Syntax: ⎕NFE 'GetLastAccessTimeUtc' filePath
Action: Returns the ⎕TS-format UTC time of the last access of the file specified by filePath

Example:
⎕NFE 'GetLastAccessTimeUtc' 'c:\myFile.ext'

GetLastWriteTime

Syntax: ⎕NFE 'GetLastWriteTime' filePath
Action: Returns the ⎕TS-format time of the last write of the file specified by filePath

Example:
⎕NFE 'GetLastWriteTime' 'c:\myFile.ext'

GetLastWriteTimeUtc

Syntax: ⎕NFE 'GetLastWriteTimeUtc' filePath
Action: Returns the ⎕TS-format UTC time of the last write of the file specified by filePath

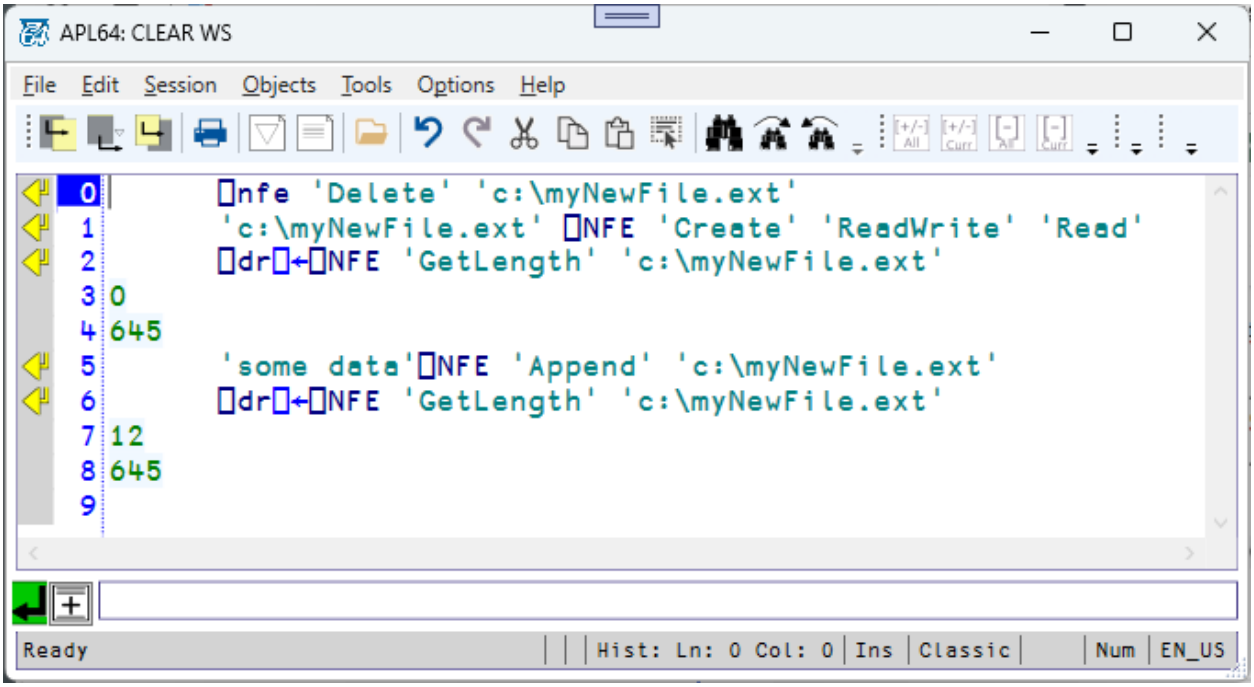
Example:
⎕NFE 'GetLastWriteTimeUtc' 'c:\myFile.ext'

GetLength

Syntax: ⎕NFE 'GetLength' filePath
Action: Returns the length of the file in bytes. The length of a file can exceed the maximum value of an integer, so the result is a scalar double. The result is not the file size on disk.

Example:

```
⎕nfe 'Delete' 'c:\myNewFile.ext'
'c:\myNewFile.ext' ⎕NFE 'Create' 'ReadWrite' 'Read'
⎕dr⎕←⎕NFE 'GetLength' 'c:\myNewFile.ext'
'some data'⎕NFE 'Append' 'c:\myNewFile.ext'
⎕dr⎕←⎕NFE 'GetLength' 'c:\myNewFile.ext'
```

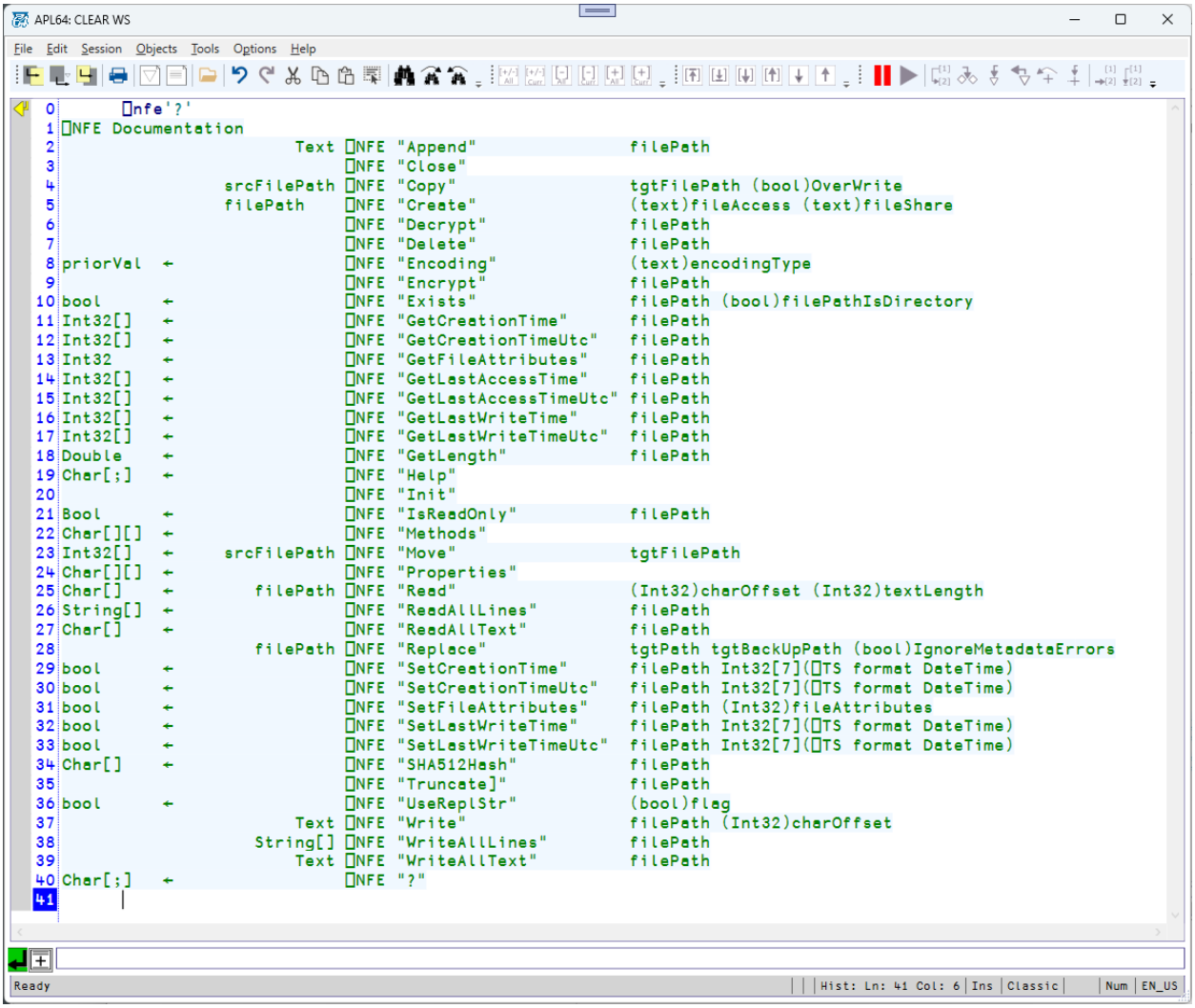


Help

Syntax: ⎕NFE 'Help' or ⎕NFE '?'

Action: Returns a text matrix containing a summary of ⎕NFE.

Example: ⎕NFE 'Help'



Init

This is an APL+Win]NFE action which is deprecated in APL64.

Move

Syntax: srcPath]NFE 'Move' tgtPath

Result: None

Action:

Moves the file specified by srcPath to a the file specified by tgtPath

Throws an exception if the tgtPath file exists

Can be used to rename a file

If the file specified by srcPath is tied in APL64, this method closes that tie

Examples:

'c:\myFile1.ext']NFE 'Move' 'e:\myFile1.exe'

'c:\myFile1.ext']NFE 'Move' 'e:\myFile1Moved.ext'

'c:\myFile1.ext']NFE 'Move' 'c:\myFile1NewName.ext'

Read

Syntax: filePath ⎕NFE 'Read' charOffset textLength
Return: text vector
Action:
Read the encoding from the file and returns the associated text in the file specified by filePath
Starting at charOffset (index origin zero) in the file
For a length of textLength
The NFE encoding property affects this method
The NFE Read method supports random-access reading of text from an NFE file.
The NFE Read method will read only up to the end of the file if the charOffsetand textLength arguments point to a location beyond the end of the file.

Examples:
⎕NFE 'encoding' 'Unicode'
'c:\myFile.ext' ⎕NFE 'Read' 100 5000

ReadAllLines

Syntax: string[]←⎕NFE 'ReadAllLines' filePath
Return: vector of strings
Action:
Read the encoding from the file and returns the associated text in the file specified by filePath with one string in the result vector for each line of text in the file.
The NFE encoding property affects this method

Examples:
⎕NFE 'encoding' 'Unicode'
sv← ⎕NFE 'ReadAllLines' 'c:\myFile.ext' ⍝ Vector of strings result
ca←⊃>sv ⍝Character array

ReadAllText

Syntax: char[]←⎕NFE 'Read' filePath
Return: Text vector
Action:
Read the encoding from the file and returns the associated text in the file specified by filePath
The NFE encoding property affects this method

Examples:
⎕NFE 'encoding' 'Unicode'
⎕NFE 'ReadAllText' 'c:\myFile.ext'

Replace

Syntax: srcPath ⎕NFE 'Replace' tgtPath tgtBackUpPath boolIgnoreMetadataErrors
Action:

Copies the file specified by tgtPath to the file specified by the tgtBackUpPath
Replaces the contents of the file specified by tgtPath with the contents of the file specified by srcPath
boolIgnoreMetadataError: 0/False 1/True
Result: None

Example:
'c:\sourceFile.ext' ⎕NFE 'Replace' 'x:\overwrittenfile.exe' 'd:\backupFile.ext'

SetCreationTime

Syntax: ⎕NFE 'SetCreationTime' filePath Int32[7](⎕TS format DateTime)
Action: Sets the ⎕TS-format time of creation of the file specified by filePath
Example:
⎕NFE 'SetCreationTime' 'c:\myFile.ext' ⎕TS

SetCreationTimeUtc

Syntax: ⎕NFE 'SetCreationTimeUtc' filePath Int32[7](⎕TS format DateTime)
Action: Sets the ⎕TS-format UTC time of creation of the file specified by filePath

Example:
⎕NFE 'SetCreationTimeUTC' 'c:\myFile.ext' ⎕TS

SetFileAttributes

Syntax: ⎕NFE 'SetFileAttributes' filePath (Int32)fileAttributes
Action: Sets the .Net FileAttributes of the file specified by filePath

Example:
⎕NFE 'SetFileAttributes' 'c:\myFile.ext' 32
Use the numeric value of the FileAttributes enumeration. For additional documentation refer to: FileAttributes Enum (System.IO) | Microsoft Docs.

SetLastAccessTime

Syntax: ⎕NFE 'SetLastAccessTime' filePath Int32[7](⎕TS format DateTime)
Action: Sets the ⎕TS-format time of the last access of the file specified by filePath

Example:
⎕NFE 'SetLastAccessTime' 'c:\myFile.ext' ⎕TS

SetLastAccessTimeUtc

Syntax: ⎕NFE 'SetLastAccessTimeUtc' filePath Int32[7](⎕TS format DateTime)
Action: Sets the ⎕TS-format UTC time of the last access of the file specified by filePath

Example:
⎕NFE 'SetLastAccessTimeUtc' 'c:\myFile.ext' ⎕TS

SetLastWriteTime

Syntax: ⎕NFE 'SetLastWriteTime' filePath Int32[7](⎕TS format DateTime)
Action: Set the ⎕TS-format time of the last write of the file specified by filePath

Example:
⎕NFE 'SetLastWriteTime' 'c:\myFile.ext' ⎕TS

SetLastWriteTimeUtc

Syntax: ⎕NFE 'SetLastWriteTimeUtc' filePath Int32[7](⎕TS format DateTime)
Action: Set the ⎕TS-format UTC time of the last write of the file specified by filePath

Example:
⎕NFE 'SetLastWriteTimeUtc' 'c:\myFile.ext' ⎕TS

SHA512Hash

Syntax: ⎕NFE 'SHA512Hash' filePath
Action: Obtains the SHA-512 hash, as a character vector with dashes (-) elided, of the content of the file specified by filePath. The SHA-512 hash of two different file contents could be the same (collision) with very low probability for the hardware which runs APL64.

Example:
⎕NFE 'SHA512HASH' 'c:\junk\PersonEinstein.xml'

Truncate

Syntax: ⎕NFE 'Truncate' filePath
Action: Clears the files contents so that it contains zero bytes

Example:
⎕NFE 'Truncate' 'c:\myFile.ext'

Write

Syntax: text ⎕NFE 'Write' filePath charOffset
Action: Writes encoding of text to the file specified by filePath
The file must exist. Use the Append action if the file does not exist.
Writing starts at charOffset (index origin zero) in the file.
The NFE encoding property affects this method
For some encoding options the number of bytes per character varies. For example, for the UTF8 encoding, the encoding of an ASCII character is one byte, but the encoding of a non-ASCII character may require two, four or more bytes.

For certain encoding options, the performance of the random-access NFE Write method operation will depend strongly on the size of the file and the charOffset argument value.

A random-access Write into an encoded text file cannot simply inject the encoding of the new text into the location of the encoding of the text being replaced. This is because the number of bytes required for the encoding of the new 'text' might be greater than the number of bytes used for the encoding of the text being replaced.

The NFE Write method performs the following process to complete its action:

- Create a temporary file
- Read the encoding of the text from the source file which is before the charOffsetspecified character
- Append the encoding of this text to the temporary file
- Append the encoding of the 'text' argument of the NFE Write method to the temporary file
- Read the encoding of the remaining text from the source file, if any, which is beyond the area affected by the NFE Write method
- Append the encoding of the remaining text to the temporary file
- Replace the contents of the source file with the contents of the temporary file, creating a backup file
- Delete the backup file if not necessary
- Delete the temporary file

Result: None

Examples:

```
□NFE 'encoding' 'ASCII'
'abcde' □NFE 'Write' 'c:\myASCIIFile.txt' 0
```

The 'abcde' text will be encoded as ASCII and written to the specified file starting at character 0.

```
□NFE 'encoding' 'Unicode'
'abcde' □NFE 'Write' 'c:\myUnicodeFile.txt' 100
```

The 'abcde' text will be encoded as Unicode and written to the specified file starting after the 100th existing character in the file, if any.

WriteAllLines

Syntax: String[] □NFE 'WriteAllLines' filePath

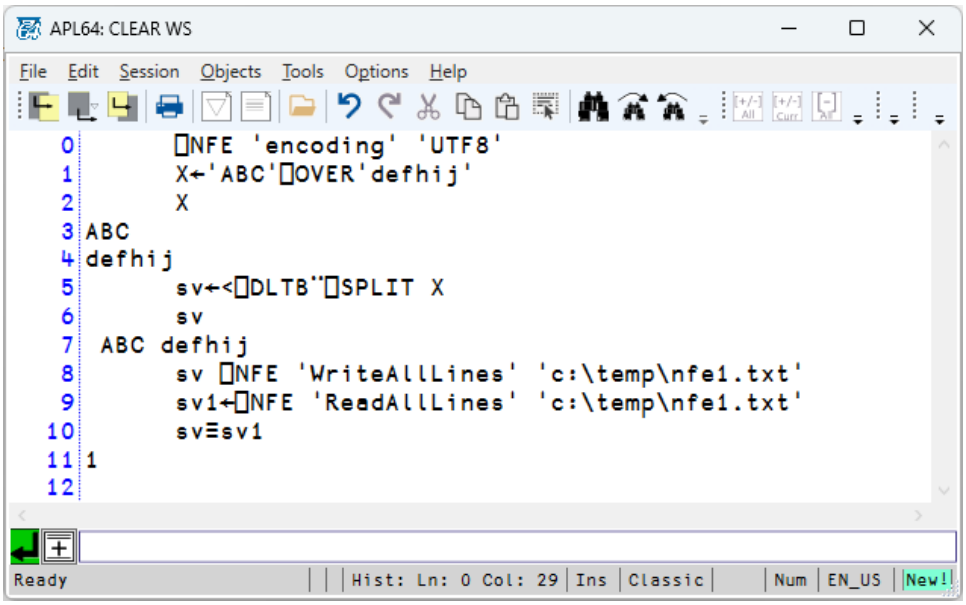
Action: Writes encoding of text lines to the file specified by filePath.

The NFE encoding property affects this method.

The target file will be overwritten if it exists.

Example:

```
□NFE 'encoding' 'UTF8'
 X←'ABC'□OVER'defhij'
sv←<□DLTB¨□SPLIT X
sv □NFE 'WriteAllLines' 'c:\temp\nfe1.txt'
sv1←□NFE 'ReadAllLines' 'c:\temp\nfe1.txt'
sv≡sv1
```



WriteAllText

Syntax: text ⎕NFE 'WriteAllLines' filePath
Action: Writes encoding of text to the file specified by filePath.
The NFE encoding property affects this method.
The target file will be overwritten if it exists.

Example:
⎕NFE 'encoding' 'UTF8'

'ABCdefhij'⎕NFE 'WriteAllText' 'c:\temp\nfe1.txt'

Example which Combines ⎕NFE Actions

In this example a text file is created, the encoding of text is written to the file and text is read from the encoding in the file.

⎕NFE 'DELETE' «c:\\text» 1 1
⍝↑ Delete the target folder and all of its sub-folders, if any

filePath ← 'c:\text\test nfe.txt'
⍝↑ Define the desired file name

⎕MKDIR 'c:\text'
⍝↑ Create the folder

⎕NFE 'Delete' filePath
⍝↑ Delete any previous instance of the filePath file

filePath ⎕NFE 'Create' 'ReadWrite' 'Read'
⍝↑ Create the filePath file with ReadWrite access for the creator and Read access for other users
⍝ Any method, including the APL64 ⎕XNCREATE system function can be used to create the file

⎕NFE 'encoding' 'UNICODE'
⍝↑ Select the Unicode encoding for this file

'⍒^Δ∇⍴÷⌸©' ⎕NFE 'Append' filePath
⍝↑ Append the encoding of the text 'AB' to the file

S ← filePath ⎕NFE 'Read' 1 2
⍝↑ S will contain the 2nd and 3rd characters in the file, '^Δ'

S ← filePath ⎕NFE 'Read' 0 1E6
⍝↑ S will contain all the characters in the file after the 1st character, '⍒^Δ∇⍴÷⌸©'

'XY' ⎕NFE 'Write' filePath 1
⍝↑ Write the encoding of the characters 'XY' to the file overwriting the pre-existing 2nd and 3rd characters

S ← ⎕NFE 'Read' filePath 0 1E6
⍝↑ S will contain the characters '⍒XY∇⍴÷⌸©'

T ← ⎕NFE 'GetCreationTime' filePath
T ← ⎕NFE 'GetCreationTimeUtc' filePath
T ← ⎕NFE 'GetLastAccessTime' filePath
T ← ⎕NFE 'GetLastAccessTimeUtc' filePath
T ← ⎕NFE 'GetLastWriteTime' filePath
T ← ⎕NFE 'GetLastWriteTimeUtc' filePath
⍝↑ Obtain various file times in various APL64 ⎕TS format