OES Cross-Platform Libraries (XPlat) for Linux
Developer Kit

January 2016
Legal Notices

For information about legal notices, trademarks, disclaimers, warranties, export and other use restrictions, U.S. Government rights, patent policy, and FIPS compliance, see https://www.novell.com/company/legal/.
Contents

About This Guide .................................................. 5

1 Overview .......................................................... 7
  1.1 Installing ...................................................... 7
    1.1.1 Linux Not Installed ...................................... 7
    1.1.2 Linux Installed .......................................... 7
  1.2 Configuring .................................................. 8
  1.3 Compiling and Building ..................................... 8
  1.4 Troubleshooting ............................................ 8
  1.5 Limitations .................................................. 9
    1.5.1 System Code Pages ...................................... 9
    1.5.2 UTF-8 Names ........................................... 9
  1.6 Code Pages .................................................. 9

2 Functions ........................................................ 11
  NWCommitFile ............................................... 12
  NWGetFilePos ............................................... 13
  NWGetFileSize ............................................... 14
  NWReadFile ................................................ 15
  NWSetFilePos ............................................... 16
  NWSetFileSize ............................................... 17
  NWWriteln .................................................. 18

3 Thread Session Functions ....................................... 19
  NWCreateThreadSessionContext ................................ 20
  NWDestroyThreadSessionContext ................................ 21
  NWSetThreadSessionContext ................................... 22
  NWGetThreadSessionContext ................................... 23
  NWGetThreadSessionContextCount ................................ 24
  NWClearThreadSessionContext .................................. 25

4 Greater Than 16 TB Volume Support Functions .................... 27
  NWGetDirSpaceInfoExt ......................................... 28
  NWGetExtendedVolumeInfoExt .................................. 30
  NWGetVolumeDetailsByInfoMask ................................ 32
  NWGetObjDiskRestrictionsExt .................................. 34
  NWGetDirSpaceLimit .......................................... 36
  NWSetDirSpaceLimitExt ....................................... 38
  NWSetObjectVolSpaceLimitExt ................................ 40
  NWScanVolDiskRestrictionsExt .................................. 42

A Documentation Revision History ................................ 45
About This Guide

OES Cross-Platform Libraries (XPlat) for Linux contains the linker and header files that you need for Novell® related application development on Linux*. Applications using XPlat don’t need the Novell Client™ for Linux. However, certain operations (such as mapping network drives) won’t work unless you install the Novell Client for Linux.

This guide contains the following sections:

- Chapter 1, “Overview,” on page 7
- Chapter 2, “Functions,” on page 11
- Chapter 3, “Thread Session Functions,” on page 19
- Chapter 4, “Greater Than 16 TB Volume Support Functions,” on page 27
- Appendix A, “Documentation Revision History,” on page 45

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Documentation Updates

For the most recent version of this guide, see OES Cross-Platform Libraries (XPlat) for Linux (https://www.novell.com/developer/ndk/xplat-linux.html).

Additional Information

For more Linux functions, see OES Cross-Platform Libraries (XPlat) for Linux (https://www.novell.com/developer/ndk/xplat-linux.html).

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items within a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.
1 Overview

OES Cross-Platform Libraries (XPlat) for Linux contains all of the required linker and header files that you need to develop Novell® related applications on Linux.

You don’t need to install the Novell Client™ for Linux. However, some operations (such as mapping network drives) won’t work unless the Novell Client for Linux is installed.

If you already have the Novell Client for Linux or Open Enterprise Server 2015 (OES 2015) installed, you only need to install the devel and header packages from XPlat for application development.

1.1 Installing

What you need to install depends on what you already have installed:

- Section 1.1.1, "Linux Not Installed," on page 7
- Section 1.1.2, "Linux Installed," on page 7

1.1.1 Linux Not Installed

If OES 2015 or the Novell Client for Linux aren’t already installed, install the following packages:

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>novell-xplatlib-devel</td>
<td>Linker files</td>
</tr>
<tr>
<td>novell-xplatlib-headers</td>
<td>SDK header files</td>
</tr>
<tr>
<td>novell-xplatlib</td>
<td>XPlat library files</td>
</tr>
<tr>
<td>novell-xtier-xplat</td>
<td>XTier requester module, xtxplat.so</td>
</tr>
<tr>
<td>novell-xtier-base</td>
<td>XTier base module</td>
</tr>
<tr>
<td>novell-xtier-core</td>
<td>XTier main module</td>
</tr>
<tr>
<td>novell-nmasclient</td>
<td>NMAS™ authentication modules</td>
</tr>
<tr>
<td>nici or nici64</td>
<td>NICI Crypto services</td>
</tr>
</tbody>
</table>

1.1.2 Linux Installed

If OES 2015 or the Novell Client for Linux are already installed, you need to install only the following two packages:
1.2 Configuring

If the Novell Linux Client isn’t installed, users other than the root user need to be added to the novlxtier group for XPlat to work. If the client is not installed, XPlat uses XTier’s requester interface module. Otherwise, XPlat requests are routed through the client’s requester interface in novfs to novfsd, which runs as root and interfaces with XTier.

Public connections that are created through the client have user scope, which means that all processes for a given user share the same connection to a server. Without the client, connections have process scope and multiple private connections from the same process can be created.

Currently, we recommend using XPlat without the client. If you need to use the client, you can temporarily disable it by entering `rcnovfsd stop` at a console, which allows you to test your application without the client running.

1.3 Compiling and Building

By default, the XPlat header files are downloaded to `/opt/novell/include/xplat`, and the XPlat libraries are downloaded to `/opt/novell/lib64` and are named as follows:

- cllnlx
- ncplnx
- callnx
- loclnx
- netlnx
- clxlnx

XPlat uses Unicode* UCS-2, not UCS-4, so the GCC `-fshort-wchar` compile flag must be used for Unicode strings.

1.4 Troubleshooting

To troubleshoot XTier, make sure `novell-xregd` is running (`rcnovell-xregd start or stop`) and that `/var/opt/novell/xtier/xregd` and `/var/opt/novell/xtier` are owned by novlxregd with the novlxtier group.

Also, see the note about other users needing to be added to the novlxtier group in Section 1.2, “Configuring,” on page 8.
1.5 Limitations

There are a few limitations in using the OES Cross-Platform Libraries (XPlat) (https://www.novell.com/developer/ndk/xplat-linux.html) functions on Linux:

- Section 1.5.1, “System Code Pages,” on page 9
- Section 1.5.2, “UTF-8 Names,” on page 9

1.5.1 System Code Pages

On Linux, the Unicode NWUS and NWUX conversion functions (http://developer.novell.com/documentation/clib/ucod_enu/data/h7qwv271.html) support only the 0 system code page. (Usually, the system code page is UTF-8 on Linux.)

The system code page is determined by the current locale character type, LC_CTYPE, which is set by the NWLsetlocale function (http://developer.novell.com/ndk/doc/clib/intl_enu/data/sdk633.html).

NWLsetlocale calls the C library setlocale function to do most of its work. However, NWLsetlocale on Linux does not accept locale strings such as 437 or 932. Instead, they must be translated to en_US.IBM437 and ja_JP.SHIFT-JIS.

NWLsetlocale was based on older UNIX* functions that did not specify language and territory. In the future, we might rework the NWUX functions to use the iconv function so that you can specify other server code pages. Also, we might change it to be more consistent with the way the binaries work on other platforms.

1.5.2 UTF-8 Names

If a volume (such as a legacy volume) does not support UTF-8, the extended file and path functions that used UTF-8 names in the past now use the non-UTF-8 functions that return names in the server’s code page. When this switch happens, all path and file names are converted to UTF-8, using the NWUX functions and the system code page.

Currently, the only way to control this behavior is to call NWLsetlocale (http://developer.novell.com/ndk/doc/clib/intl_enu/data/sdk633.html). In the future, we might add a function that sets a default OEM code page for the extended functions or allows them to return an error if UTF-8 is not supported.

1.6 Code Pages

The following table shows the compatibility between OES and Linux code pages:

<table>
<thead>
<tr>
<th>OES</th>
<th>ICONV</th>
<th>Locale</th>
</tr>
</thead>
<tbody>
<tr>
<td>437</td>
<td>United States English</td>
<td>IBM437</td>
</tr>
<tr>
<td>850</td>
<td>Multilingual</td>
<td>IBM850</td>
</tr>
<tr>
<td>852</td>
<td>Slavic</td>
<td>IBM852</td>
</tr>
<tr>
<td>855</td>
<td>Cyrillic</td>
<td>IBM855</td>
</tr>
<tr>
<td>OES</td>
<td>ICONV</td>
<td>Locale</td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>857</td>
<td>Turkish</td>
<td>IBM857</td>
</tr>
<tr>
<td>860</td>
<td>Portugese</td>
<td>IBM860</td>
</tr>
<tr>
<td>861</td>
<td>Icelandic</td>
<td>IBM861</td>
</tr>
<tr>
<td>862</td>
<td>Hebrew</td>
<td>IBM862</td>
</tr>
<tr>
<td>863</td>
<td>Canadian French</td>
<td>IBM863</td>
</tr>
<tr>
<td>865</td>
<td>Nordic</td>
<td>IBM865</td>
</tr>
<tr>
<td>866</td>
<td>Russian</td>
<td>IBM866</td>
</tr>
<tr>
<td>874</td>
<td>Thai</td>
<td>IBM874</td>
</tr>
<tr>
<td>932</td>
<td>Japanese</td>
<td>SHIFT_JIS</td>
</tr>
<tr>
<td>936</td>
<td>Simplified Chinese</td>
<td>GBK</td>
</tr>
<tr>
<td>949</td>
<td>Korean</td>
<td>IBM949</td>
</tr>
<tr>
<td>950</td>
<td>Traditional Chinese</td>
<td>BIG5</td>
</tr>
</tbody>
</table>
Functions

These functions are new and are not included in OES Cross-Platform Libraries (XPlat) for Linux (https://www.novell.com/developer/ndk/xplat-linux.html).

The following file access functions are available to access OES files on Linux:

- “NWCommitFile” on page 12
- “NWGetFilePos” on page 13
- “NWGetFileSize” on page 14
- “NWReadFile” on page 15
- “NWSetFilePos” on page 16
- “NWSetFileSize” on page 17
- “NWWriteFile” on page 18
**NWCommitFile**

Flushes a written file from a cache to disk.

**Syntax**

```c
#include <nwfile.h>

N_EXTERN_LIBRARY (NWCCODE) NWCommitFile(
    NWFILE_HANDLE   fileHandle);
```

**Parameters**

*fileHandle*

(IN) Specifies the file to commit.

**Return Values**

For more information, see the Return Values for C documentation (http://developer.novell.com/documentation/general/genl_enu/data/bktitle.html).
NWGetFilePos

Returns the position of a file.

Syntax

```c
#include <nwfile.h>

N_EXTERN_LIBRARY (NWCCODE) NWGetFilePos(
    NWFILE_HANDLE   fileHandle,
    pnumint64       filePos);
```

Parameters

fileHandle

(IN) Specifies the file to return the position for.

filePos

(OUT) Points to the position of the file.

Return Values

For more information, see the Return Values for C documentation (http://developer.novell.com/documentation/general/genl_enu/data/bktitle.html).
NWGetFileSize

Returns the size of a file.

Syntax

```
#include <nwfile.h>

N_EXTERN_LIBRARY (NWCCODE) NWGetFileSize(
    NWFILE_HANDLE   fileHandle,
    pnuint64        fileSize);
```

Parameters

fileHandle

(IN) Specifies the file to return a size for.

fileSize

(OUT) Points to the size of the file.

Return Values

For more information, see the Return Values for C documentation (http://developer.novell.com/documentation/general/genl_enu/data/bkttitle.html).
NWReadFile

Reads a file.

Syntax

#include <nwfile.h>

N_EXTERN_LIBRARY (NWCCODE) NWReadFile(
   NWFILE_HANDLE   fileHandle,
   nuint32         bytesToRead,
   puint32        bytesRead,
   puint8         data);

Parameters

fileHandle
   (IN) Specifies the file to read.

bytesToRead
   (IN) Specifies the number of bytes to read.

bytesRead
   (OUT) Specifies the number of bytes that were read.

data
   (OUT) Points to the data that was read.

Return Values

For more information, see the Return Values for C documentation (http://developer.novell.com/documentation/general/genl_enu/data/bkttitle.html).
**NWSetFilePos**

Sets the position of a file.

**Syntax**

```c
#include <nwfile.h>

N_EXTERN_LIBRARY (NWCCODE) NWSetFilePos(
    NWFILE_HANDLE   fileHandle,
    nuint           mode,
    nint64          filePos);
```

**Parameters**

- **fileHandle**
  
  (IN) Specifies the file to set the position for.

- **mode**
  
  (IN) Specifies the position mode of the file:
  
  0 SEEK_FROM_BEGINNING
  1 SEEK_FROM_CURRENT
  2 SEEK_FROM_END

- **filePos**
  
  (IN) Specifies the position of the file.

**Return Values**

For more information, see the Return Values for C documentation (http://developer.novell.com/documentation/general/genl_enu/data/bktitle.html).
**NWSetFileSize**

Sets the size of a file.

**Syntax**

```
#include <nwfile.h>

N_EXTERN_LIBRARY (NWCCODE) NWSetFileSize(
    NWFILE_HANDLE   fileHandle,
    nuint64         fileSize);
```

**Parameters**

- **fileHandle**
  - (IN) Specifies the file to set the size for.

- **fileSize**
  - (IN) Specifies the size of the file.

**Return Values**

For more information, see the Return Values for C documentation (http://developer.novell.com/documentation/general/genl_enu/data/bktitle.html).
**NWWriteFile**

Writes a file.

**Syntax**

```
#include <nwfile.h>

N_EXTERN_LIBRARY (NWCCODE) NWWriteFile(
    NWFILE_HANDLE   fileHandle,
    nuint32         bytesToWrite,
    puint8         data);
```

**Parameters**

- **fileHandle**
  
  (IN) Specifies the file to write to.

- **bytesToWrite**
  
  (IN) Specifies the number of bytes to write.

- **data**
  
  (OUT) Points to the data that was written.

**Return Values**

For more information, see the Return Values for C documentation (http://developer.novell.com/documentation/general/genl_enu/data/bktitle.html).
Thread Session Functions

The Thread Session Context functions are available on the Linux platform. Using these functions you can migrate OES applications that use XPlat libraries to Linux platform.

An application can manage user identities and connections using these functions.

Using these functions an application can create new XTier session contexts and assigns them to various threads. Connections and user identities are associated with a session context. Using a unique session context, a thread can create its own private connections and identities. You can also share session contexts with threads. A thread uses the default session context if it is not assigned to a session context handle.

XPlat uses OES Requester API to perform the basic functions like creating connections, authenticate users etc. By default, XPlat uses Novell Client’s Requester interface. If Novell Client is not installed, XPlat uses XTier’s requester interface module. The client shares the connections and identities between the user processes. Even if Novell Client is installed, you can force an application to use XTier by setting an environment variable to “XPLAT_USE_XTIER”.

NOTE: For the thread session functions to work, XTier’s requester interface must be available.

The following thread session context functions are available only on Linux:

- “NWCreateThreadSessionContext” on page 20
- “NWDestroyThreadSessionContext” on page 21
- “NWSetThreadSessionContext” on page 22
- “NWGetThreadSessionContext” on page 23
- “NWGetThreadSessionContextCount” on page 24
- “NWClearThreadSessionContext” on page 25

If XTier interface is not available, the functions returns NWE_REQUESTER_FAILURE. Also the XPlat initialization functions like NWCallsInit(), NWCLXInit, NWNetInit fails to succeed if the XTier interface is not available.
**NWCreateThreadSessionContext**

Creates a new session context.

**Syntax**

```c
#include <nwmisc.h>
N_EXTERN_LIBRARY(NWCODE) NWCreateThreadSessionContext (TSCHANDLE *phTSC);
```

**Parameters**

*phTSC*  
(OUT) Points to a new session context handle.

**Return Values**

0x0000 SUCCESS  
0x881A NWE_OUT_OF_HEAP_SPACE  
0x8836 NWE_PARAM_INVALID  
0x88FF NWE_REQUESTER_FAILURE
NWDestroyThreadSessionContext

Destroys a created session context.

Syntax

```c
#include <nwmisc.h>
N_EXTERN_LIBRARY(NWCCODE) NWDestroyThreadSessionContext (TSCHANDLE hTSC);
```

Parameters

- **hTSC**
  - (IN) Specifies the session context handle.

Return Values

- 0x0000 SUCCESS
- 0x8836 NWE_PARAM_INVALID
- 0x88FF NWE_REQUESTER_FAILURE

Remarks

Destroying a session context handle automatically removes the handle from any threads that are assigned to it, causing them to revert to the global default session context.
**NWSetThreadSessionContext**

Assigns a session context to a thread.

**Syntax**

```c
#include <nwmisc.h>
N_EXTERN_LIBRARY(NWCCODE) NWSetThreadSessionContext (TSCHANDLE hTSC);
```

**Parameters**

- **hTSC**
  - (IN) Specifies the session context handle.

**Return Values**

- 0x0000 SUCCESS
- 0x8800 NWE_ALREADY_ATTACHED
- 0x8836 NWE_PARAM_INVALID
- 0x88FF NWE_REQUESTER_FAILURE

**Remarks**

The return value NWE_ALREADY_ATTACHED indicates that a thread is already assigned to a session context which needs to be cleared from this thread for this function to succeed.
NWGetThreadSessionContext

Returns a thread’s assigned session context.

Syntax

```
#include <nwmisc.h>
N_EXTERN_LIBRARY(NWCCODE) NWGetThreadSessionContext (TSCHANDLE *phTSC);
```

Parameters

`phTSC`

(OUT) Points to a session context handle.

Return Values

0x0000 SUCCESS
0x8836 NWE_PARAM_INVALID
0x886F NWE_OBJECT_NOT_FOUND
0x88FF NWE_REQUESTER_FAILURE

Remarks

If there is no session context assigned to the thread, NWE_OBJECT_NOT_FOUND is returned and *phTSC is set to NULL.
NWGetThreadSessionContextCount

Returns the number of threads assigned to the session context.

Syntax

```c
#include <nwmisc.h>
N_EXTERN_LIBRARY(NWCCODE) NWGetThreadSessionContextCount (TSCHANDLE hTSC, pnuint32 count);
```

Parameters

- **hTSC**
  - (IN) Specifies the session context handle.

- **count**
  - (OUT) Reference count

Return Values

- 0x0000 SUCCESS
- 0x8836 NWE_PARAM_INVALID
- 0x88FF NWE_REQUESTER_FAILURE

Remarks

Helper function.
NWClearThreadSessionContext

Removes a thread’s assigned session context.

Syntax

# include <nwmisc.h>
N_EXTERN_LIBRARY(NWCCODE) NWClearThreadSessionContext (void);

Return Values

0x0000 SUCCESS
0x886F NWE_OBJECT_NOT_FOUND
0x88FF NWE_REQUESTER_FAILURE

Remarks

When a thread's session context is removed, the thread reverts to the global default session context. If a thread is not assigned any session context, NWE_OBJECT_NOT_FOUND is returned.
The NCP protocol has been extended to support volumes larger than 16 TB. Using these functions you can obtain the volume information and other size details through NCP, which supports 64-bit volume size information.

The following volume support functions are available on Linux:

- “NWGetDirSpaceInfoExt” on page 28
- “NWGetExtendedVolumeInfoExt” on page 30
- “NWGetVolumeDetailsByInfoMask” on page 32
- “NWGetObjDiskRestrictionsExt” on page 34
- “NWGetDirSpaceLimit” on page 36
- “NWSetDirSpaceLimitExt” on page 38
- “NWSetObjectVolSpaceLimitExt” on page 40
- “NWScanVolDiskRestrictionsExt” on page 42
NWGetDirSpaceInfoExt

Returns information on space usage for a volume or directory.

**Remote Servers:** blocking

**OES Server:** OES 2015.0

**Platform:** Windows 7 or later, Linux

**Library:** Cross-Platform Calls (CAL.*)

**Service:** File System

### Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE ) NWGetDirSpaceInfoExt( 
    NWCONN_HANDLE    conn,
    NWDIR_HANDLE     dirHandle,
    nuint32         volNum,
    DIR_SPACE_INFO2 N_FAR * spaceInfo);
```

### Parameters

- **conn**
  - (IN) Specifies the OES server connection handle.

- **dirHandle**
  - (IN) Specifies the directory handle associated with the desired directory path (0 if volume information is to be returned).

- **volNum**
  - (IN) Specifies the volume number to return space information for (0 if directory information is to be returned).

- **spaceInfo**
  - (OUT) Points to the DIR_SPACE_INFO2 structure.

### Return Values

- 0x0000  SUCCESSFUL
- 0x899B  BAD_DIRECTORY_HANDLE
- 0x8998  VOLUME_DOES_NOT_EXIST
- 0x89FD  BAD_STATION_NUMBER
- 0x89FF  FAILURE (Bad Info Type or Bad return info mask)
Remarks

If the dirHandle parameter is zero, NWGetDirSpaceInfoExt returns the volume information to the DIR_SPACE_INFO2 structure. Pass the volume number in volNum, which is obtained from calling NWGetVolumeNumber.

purgeableBlocks and nonYetPurgeableBlocks are set to 0 if the dirHandle parameter contains a nonzero value.

The availableBlocks field is the only field that returns information when disk space restrictions are in effect. The rest of the structure fields contain volume-wide information. If disk space restrictions are not in effect, the availableBlocks field will contain the number of blocks available for use on the entire volume.

One block equals the size of the block size for the specified volume, which is obtained by multiplying sectorsPerBlock by 512 bytes.

You can call NWGetExtendedVolumeInfoExt (Volume Services) to return the block size (in bytes).

NCP Calls

0x2222 123 35 Get Volume Purge Information

0x2222 22 58 Get Dir Info
**NWGetExtendedVolumeInfoExt**

Returns extended volume information.

**Remote Servers**: blocking

**OES Server**: OES 2015.0

**Platform**: Windows 7 or later, Linux

**Library**: Cross-Platform Calls (CAL*.*)

**Service**: Volume

**Syntax**

```c
#include <nwvol.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE ) NWGetExtendedVolumeInfoExt(  
    NWCONN_HANDLE   conn,  
    nuint32         volNum,  
    NWVolExtendedInfo2 N_FAR * volInfo);
```

**Parameters**

- **conn**
  - (IN) Specifies the OES server connection handle.

- **volNum**
  - (IN) Specifies the volume number.

- **volInfo**
  - (OUT) Points to NWVolExtendedInfo2, which receives information.

**Return Values**

- 0x0000  SUCCESSFUL
- 0x8998  VOLUME_DOES_NOT_EXIST
- 0x899B  BAD_DIRECTORY_HANDLE
- 0x89FD  BAD_STATION_NUMBER
- 0x89FB  FAILURE (Bad Info Type or Bad return info mask)

**Remarks**

NWGetExtendedVolumeInfoExt returns information based on the volume block size (64 KB), which can be determined using the formula:

\[(\text{sectorSize} \times \text{sectorsPerCluster}) / 1024\]

NWGetExtendedVolumeInfoExt must be called for a licensed connection or FAILURE will be returned.
NCP Calls

0x2222 123 35 Get Extended Volume Information
NWGetVolumeDetailsByInfoMask

Returns information for the specified volume based on the information structure provided in ReturnInfoMask.

Remote Servers: blocking

OES Server: OES 2015.0

Platform: Windows 7 or later, Linux

Library: Cross-Platform Calls (CAL*.*)

Service: Server Environment

Syntax

```c
#include <nwfse.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE ) NWGetVolumeDetailsByInfoMask( 
    NWCONN_HANDLE   conn, 
    nuint32         volNum, 
    nuint32         dirHandle, 
    nuint32         ReturnInfoMask, 
    NWFSE_VOLUME_DETAILS_BY_INFOMASK N_FAR * fseVolumeDetails); 
```

Parameters

**conn**

(IN) Specifies the OES server connection handle.

**volNum**

(IN) Specifies the volume number to return space information for (0 if directory information is to be returned).

**dirHandle**

(IN) Specifies the directory handle associated with the desired directory path (0 if volume information is to be returned).

**ReturnInfoMask**

(IN) Specifies the ReturnInfoMask information to return (VINFO_RIM_VOL_INFO64 or VINFO_RIM_VOL_NAME or VINFO_RIM_VOL_INFO64|VINFO_RIM_VOL_NAME).

**fseVolumeDetails**

(OUT) Points to NWFSE_VOLUME_DETAILS_BY_INFOMASK, which receives information.

Return Values

0x0000  SUCCESSFUL
0x8998  VOLUME_DOES_NOT_EXIST
0x899B  BAD_DIRECTORY_HANDLE
0x89FD  BAD_STATION_NUMBER
0x89FF  FAILURE (Bad Info Type or Bad return info mask)
Remarks

In reply data, only the information structure indicated in ReturnInfoMask will be returned in the reply length. That is, if the request comes with only VINFO_RIM_VOL_NAME for ReturnInfoMask, the reply contains only structure VolumeNameDetails along with filled data and starts at the offset without considering the size for VolumeInfo_64.

Similarly, if the request comes with VINFO_RIM_VOL_INFO64 for ReturnInfoMask, the reply contains only structure VolumeInfoDetails. Also, if the request comes with VINFO_RIM_VOL_INFO64 | VINFO_RIM_VOL_NAME for ReturnInfoMask, the reply contains both the VolumeInfoDetails and VolumeNameDetails.

Order of the information filled by the server is in the order of it's bit mask. That is, the info with bit mask 0x00000001 is filled first (if requested), followed by the information with bit mask 0x00000002 is filled next, followed by 0x00000003 and so on.

NCP Calls

0x2222 123 35 Get Volume Information By InfoMask
**NWGetObjDiskRestrictionsExt**

Returns the disk restrictions imposed on an object for the specified volume number.

**Remote Servers:** blocking

**OES Server:** OES 2015.0

**Platform:** Windows 7 or later, Linux

**Library:** Cross-Platform Calls (CAL*.*)

**Service:** Volume

### Syntax

```c
#include <nwvol.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE ) NWGetObjDiskRestrictionsExt(
    NWCONN_HANDLE   conn,
    nuint32         volNumber,
    nuint32         objectID,
    pnuint64        restriction,
    pnuint64        inUse);
```

### Parameters

- **conn**
  - (IN) Specifies the OES server connection handle.

- **volNumber**
  - (IN) Specifies the volume number for which the restrictions has to be returned.

- **objectID**
  - (IN) Specifies the object ID.

- **restriction**
  - (OUT) Points to the buffer containing the number of blocks the object can use.

- **inUse**
  - (OUT) Points to the buffer containing the number of blocks the object is currently using.

### Return Values

- 0x0000  SUCCESSFUL
- 0x8998  VOLUME_DOES_NOT_EXIST

### Remarks

The restrictions are returned in units of 4KB blocks and ignore the block size of the volume.
NOTE: The valid restriction values are as follows:

- If the restriction equals 0x7fffffffffffffff, the object has no restrictions.
- If the restriction equals 0, the object has full restrictions or no space allowed.
- If the restriction value is from 1 to 0x7fffffffffffffff, the object has restrictions based on the corresponding value.

NCP Calls

0x2222 22 55 Get Object Disk Usage And Restrictions
NWGetDirSpaceLimit

Returns the actual space limitations for a directory.

Remote Servers: blocking

OES Server: OES 2015.0

Platform: Windows 7 or later, Linux

Library: Cross-Platform Calls (CAL.*)

Service: File System

Syntax

```c
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE ) NWGetDirSpaceLimit(
    NWCONN_HANDLE   conn,
    NWDIR_HANDLE    dirHandle,
    NW_RESTRICTION_64 N_FAR * Restrictions);
```

Parameters

conn

(IN) Specifies the OES server connection handle.

dirHandle

(IN) Specifies the directory handle pointing to the desired directory.

Restrictions

(OUT) Points to Restrictions.

Return Values

0x0000  SUCCESSFUL
0x8977  ERR_BUFFER_TOO_SMALL
0x8996  SERVER_OUT_OF_MEMORY
0x8997  ERR_TARGET_NOT_A_SUBDIRECTORY
0x8998  VOLUME_DOES_NOT_EXIST
0x899B  BAD_DIRECTORY_HANDLE
0x899C  INVALID_PATH
0x89A1  DIRECTORY_IO_ERROR
0x89BF  INVALID_NAME_SPACE
0x89FD  BAD_STATION_NUMBER
0x89FF  FAILURE (Invalid Request Parameter)
Remarks

To find the actual amount of space (MinSpaceLeft) available to a directory, scan the amount of disk space assigned to all directories between the current directory and root directory. The smallest of the SpaceLeft values for the current directory and ancestor directories in the path is calculated and returned. If the MinSpaceLeft is zero, there is no space in the directory.

All restrictions are returned in units of 4KB blocks.

The valid restriction values are as follows:

- If the restriction equals 0x7fffffffffffffff, the object has no restrictions.
- If the restriction equals 0, the object has full restrictions or no space allowed.
- If the restriction value is from 1 to 0x7fffffffffffffff, the object has restrictions based on the corresponding value.

NOTE: If you use this function in a loop on an NSS volume, server utilization can rise to 100% which causes a denial of service to connections. You need to limit the number of quick calls to this function to under 200 and then let the server utilization drop before calling another set. Server utilization is not affected by numerous quick calls to this function on traditional volumes.

NCP Calls

0x2222 89 41 Get Directory Disk Space Restriction
NWSetDirSpaceLimitExt

Specifies a space limit (in 4 KB blocks) on a particular subdirectory.

**Remote Servers:** blocking

**OES Server:** OES 2015.0

**Platform:** Windows 7 or later, Linux

**Library:** Cross-Platform Calls (CAL*)

**Service:** File System

### Syntax

```c
#include <nwdirect.h>
#include <nwcalls.h>

NWSetDirSpaceLimitExt(
    NWCONN_HANDLE   conn,
    NWDIR_HANDLE    dirHandle,
    nuint64         spaceLimit);
```

### Parameters

- **conn**
  - (IN) Specifies the OES server connection handle.

- **dirHandle**
  - (IN) Specifies the OES directory handle pointing to the directory to scan.

- **spaceLimit**
  - (IN) Specifies the directory space limit (in 4 KB sizes).

### Return Values

- 0x0000 SUCCESSFUL
- 0x8901 ERR_INSUFFICIENT_SPACE
- 0x898C NO_MODIFY_PRIVILEGES
- 0x899B BAD_DIRECTORY_HANDLE
- 0x899C INVALID_PATH
- 0x89FD BAD_STATION_NUMBER

### Remarks

The valid restriction values are as follows:

- If the restriction equals 0x7fffffffffffffff, the object has no restrictions.
- If the restriction equals 0, the object has full restrictions or no space allowed.
- If the restriction value is from 1 to 0x7fffffffffffffff, the object has restrictions based on the corresponding value.
NOTE: All restrictions are set in units of 4KB blocks.

NSS volumes and traditional volumes have very different architectures, so this function behaves differently, depending upon the volume the directory resides on. For example, traditional volumes take a long time to mount because as the volume mounts, all entries are placed in memory and disk space usage information is calculated and kept current. NSS volumes mount quickly because the entire file system is not scanned and thus disk space usage information must be calculated when a request comes in. For a few disk space requests, you will not see a great deal of difference between an NSS volume and a traditional volume. However, if you send through 3000 requests at the same time to an NSS volume, utilization can spike to 100%, causing the server to drop connections.

NCP Calls

0x2222 22 57 Set Directory Disk Space Restrictions
NWSetObjectVolSpaceLimitExt

Sets an object disk space limit on a volume.

Remote Servers: blocking

OES Server: OES 2015.0

Platform: Windows 7 or later, Linux

Library: Cross-Platform Calls (CAL*.*)

Service: Volume

Syntax

```c
#include <nwvol.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE ) NWSetObjectVolSpaceLimitExt(
    NWCONN_HANDLE   conn,
    nuint32         volNum,
    nuint32         objID,
    nuint64         restriction);
```

Parameters

- **conn**
  - (IN) Specifies the OES server connection handle.

- **volNum**
  - (IN) Specifies the volume number for which to set the space limit.

- **objID**
  - (IN) Specifies the object ID for which to limit the volume space.

- **restriction**
  - (IN) Specifies the number of blocks (in 4KB sizes) to limit the volume space.

Return Values

- **0x0000** SUCCESSFUL
- **0x898C** NO_MODIFY_PRIVILEGES
- **0x8996** SERVER_OUT_OF_MEMORY
- **0x8998** VOLUME_DOES_NOT_EXIST

Remarks

The valid restriction values are as follows:

- If the restriction equals 0xffffffffffffffff, the object has no restrictions.
• If the restriction equals 0, the object has full restrictions or no space allowed.
• If the restriction value is from 1 to 0x7fffffffffffffff, the object has restrictions based on the corresponding value.

The restrictions are returned in units of 4KB blocks.

**NCP Calls**

0x2222 22 54 Add User Disk Space Restriction
NWScanVolDiskRestrictionsExt

Returns a list of the disk restrictions for a volume.

**Remote Servers:** blocking

**OES Server:** OES 2015.0

**Platform:** Windows 7 or later, Linux

**Library:** Cross-Platform Calls (CAL.*)

**Service:** Volume

**Syntax**

```c
#include <nwvol.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE ) NWScanVolDiskRestrictionsExt(
    NWCONN_HANDLE   conn,
    nuint32         volNum,
    pnuint32        iterhandle,
    NWVOL_RESTRICTIONS_EXT N_FAR * volInfo);
```

**Parameters**

**conn**

(IN) Specifies the OES server connection handle.

**volNum**

(IN) Specifies the volume number for which to return the restrictions.

**iterhandle**

(OUT) Points to the sequence number to use in the search (set to 0 initially).

**volInfo**

(OUT) Points to NWVOL_RESTRICTIONS_EXT.

**Return Values**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>SUCCESSFUL</td>
</tr>
<tr>
<td>0x8977</td>
<td>ERR_BUFFER_TOO_SMALL</td>
</tr>
<tr>
<td>0x8998</td>
<td>VOLUME_DOES_NOT_EXIST</td>
</tr>
</tbody>
</table>
Remarks

NWScanVolDiskRestrictionsExt function uses a larger structure for the volume restrictions that allows up to 16 restrictions per volume.

The information returned in NWVOL_RESTRICITONS_EXT contains the object restrictions that have been made for the volume. All restrictions are returned in 4KB blocks. The valid restriction values are as follows:

- If the restriction equals 0x7fffffffffffffff, the object has no restrictions.
- If the restriction equals 0, the object has full restrictions or no space allowed.
- If the restriction value is from 1 to 0x7fffffffffffffff, the object has restrictions based on the corresponding value.

IMPORTANT: NWScanVolDiskRestrictionsExt is called iteratively to retrieve information on all disk space restrictions. The number of entries is returned in the volInfo.numberOfEntries field. If the volInfo.numberOfEntries field returns the value 16, then it is assumed that there are additional entries to be returned. In this case, the value of volInfo.numberOfEntries field must be added to the previous iterhandle to obtain the value for the next iterative call.

NCP Calls

0x2222 22 56 Scan Volume’s User Disk Restrictions
The following sections outline the changes that have been made to the OES Cross-Platform Libraries (XPlat) for Linux documentation (in reverse chronological order):

<table>
<thead>
<tr>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 2016</td>
<td>Added Chapter 4, &quot;Greater Than 16 TB Volume Support Functions,&quot; on page 27</td>
</tr>
<tr>
<td>October 17, 2007</td>
<td>Added to the NDK.</td>
</tr>
</tbody>
</table>