

4D Line

User Manual

Version 1.0

e-Node

30 rue de la République

33150 Cenon

France

www.e-node.net

Copyright and Trademarks

All trade names referenced in this document are the trademarks or registered trademarks of their respective holders.

4D Line is copyright e-Node SA and exclusively published worldwide by e-Node.

4th Dimension, 4D Compiler, 4D, 4D Server, 4D Client, and 4D Insider are trademarks of 4D SA.

Windows, Excel and Vista are trademarks of Microsoft Corporation.

Macintosh, MacOS and MacOS X are trademarks of Apple, Inc.

Table of Contents

Copyright and Trademarks	2
About 4D Line	5
Compatibility Information	5
Demonstration Database.....	5
Technical Support.....	5
Registration	6
License types	6
Command Descriptions and Syntax	7
Installing 4D Line	7
The 4D Line User Interface	8
Overview.....	8
Tools	9
Line/Pen Tool	9
Curve Tool.....	10
Rectangle Tool	10
Oval Tool.....	12
Eraser Tool	14
Selection Tool	14
Programming 4D Line	15
Widths	15
Colors	15
Summary of Width and Color types	16
Edit Menu and Commands	17
Edit Menu Items and Commands Behavior.....	17
Selection Tool and Specific Behaviors	18
Multiple Undos/Redos	18
Loading and Saving.....	18
Configuration Commands	19
LN_Register (registrationKey:S) → resultCode:L.....	19
LN_SetWidths (areaRef:L; lineWidth:L; outlineWidth:L; eraserWidth:L) → errorCode:L	21
LN_GetWidths (areaRef:L; lineWidth:L; outlineWidth:L; eraserWidth:L) → errorCode:L.....	21
LN_SetForegroundColor (areaRef:L; red:L; green:L; blue:L) → errorCode:L	22
LN_GetForegroundColor (areaRef:L; red:L; green:L; blue:L) → errorCode:L.....	22
LN_SetBackgroundColor (areaRef:L; red:L; green:L; blue:L) → errorCode:L	23
LN_GetBackgroundColor (areaRef:L; red:L; green:L; blue:L) → errorCode:L.....	23
LN_SetFillColor (areaRef:L; red:L; green:L; blue:L) → errorCode:L	24
LN_GetFillColor (areaRef:L; red:L; green:L; blue:L) → errorCode:L	24

Table of Contents

Saving and Loading Commands	25
LN_LoadPicture (areaRef:L; picture:P) → errorCode:L	25
LN_SavePicture (areaRef:L; picture:P) → errorCode:L.....	25
LN_LoadBlob (areaRef:L; blob:X) → errorCode:L	26
LN_SaveBlob (areaRef:L; picture:P) → errorCode:L	26
Tools Commands	27
LN_SetToolBarProperties (areaRef:L; selectedTool:L; hidden:L) → errorCode:L.....	27
LN_GetToolBarProperties (areaRef:L; selectedTool:L; hidden:L) → errorCode:L.....	28
Selection and Edit Commands	29
LN_SetSelect (areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L.....	29
LN_GetSelect (areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L	29
LN_Cut (areaRef:L) → errorCode:L	30
LN_Copy (areaRef:L) → errorCode:L.....	30
LN_Paste (areaRef:L) → errorCode:L.....	31
LN_Delete (areaRef:L) → errorCode:L.....	31
LN_SelectAll (areaRef:L) → errorCode:L	32
LN_ClearArea (areaRef:L) → errorCode:L	32
LN_Undo (areaRef:L) → errorCode:L	33
LN_Redo (areaRef:L) → errorCode:L.....	33
Drawing Commands	34
LN_DrawLine (areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L.....	34
LN_DrawFrameRect (areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L.....	34
LN_DrawFillRect (areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L.....	35
LN_DrawFrameOval (areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L	35
LN_DrawFillOval (areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L.....	36
4D Line Error Codes	37

About 4D Line

4D Line is an easy-to-use tool for implementing bitmap drawing on 4th Dimension layouts. It lets your users create and modify bitmap pictures.

4D Line can be used with just one command — no special formatting is required.

Graphic tools can be displayed if you wish to let the users create and modify their pictures with drawing functions, to rapidly implement a bitmap drawing environment.

Foreground, background and fill colors, as well as line width can be set programatically and their current settings values can be read as well.

In addition, the whole Edit menu (undo/redo , cut, copy, paste, erase, select all) can be managed through the menu or 4D Line commands. Multiple undos/redos are available.

4D Line areas and their settings can be stored in 4D blob variables or fields, and the resulting pictures can be saved in 4D picture variables or fields, as well as in external picture type documents.

Uses of 4D Line are multiple:

- Create plans
- Display statistics and graphs
- Collect signatures and handwritten notes through scanners, graphics tablets or touchscreens
- Modify external picture documents
- Modeling
- Etc.

Compatibility Information

4D Line is fully compatible with 4D/4D Server 2004 or greater (including 4D v11 SQL). It is compatible with MacOS and Windows clients.

Demonstration Database

A 4D 2004 database is provided to illustrate how to use 4D Line, with its set of tools, in a 4D layout.

Technical Support

Technical support for 4D Line will be provided electronically via e-mail or on our website. You are encouraged to use the online reporting as it will be correctly routed to the appropriate support personnel.

www.e-node.net

Registration

4D Line requires a registration key to “unlock” the product making it a full working version. Call the **LN_Register** command (see [LN_Register](#) for complete details) in the *On Startup* method.

Without the registration key, 4D Line will operate in demonstration mode during 20 minutes.

License types

Like all e-Node plug-ins, 4D Line offers six different license types. There are no such things as MacOS vs Windows or Development vs Deployment:

- **Single user license.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Standalone or Runtime. Since the registration key is linked to a specific 4D license, you need to provide the number returned by the 4D command **GET SERIAL INFORMATION** (first parameter). A new license will be provided for free at any time if you change your 4D version and/or get a new 4D registration key.
- **Small server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server up to 10 users. The registration key is linked to your 4D Server license just as above.
- **Medium server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server up with 11 to 20 users. The registration key is linked to your 4D Server license just as above.
- **Large server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server over 20 users. The registration key is linked to your 4D Server license just as above.
- **Unlimited Single User.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on as many 4D Standalone, Runtime or Engine copies that run your 4D application(s). This is a yearly license, which expires one month after the date when it is to be renewed. The expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire.**
- **Unlimited OEM.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on as many 4D Server (of any number of users), 4D Standalone, Runtime or Engine copies that run your 4D application(s). This is a yearly license, which expires one month after the date when it is to be renewed. The expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire.**

A 4D database used to retrieve your 4D serial information is available from the following link:

<http://www.e-node.net/ftp/GetSerialInfo>

Command Descriptions and Syntax

Each 4D Line command (or routine) has a syntax, or rules, that describe how to use the command in your 4D database. For each command, the name of the command is followed by the command's parameters. The parameters are enclosed in parenthesis, and separated by semicolons.

Following the command syntax description, an explanation of the command's parameters is provided. For each parameter, the type of the parameter and a description is shown. Examples are provided, showing the syntax as well as how the various commands are used together.

The first parameter for most commands is the long integer reference of the 4D Line plug-in area on the layout. This parameter is required to allow the commands to operate on the correct object.

All 4D Line routines are actually functions, which return a long integer result value.

The result is 0 or an error code (such as wrong area reference), except for [LN_Register](#) (1 = success), which is the only exception.

See the [4D Line Error codes](#) section for the list of possible error codes.

Installing 4D Line

4D Line bundle plug-in will work with MacOS and Windows deployments (you don't need separate MacOS and Windows versions).

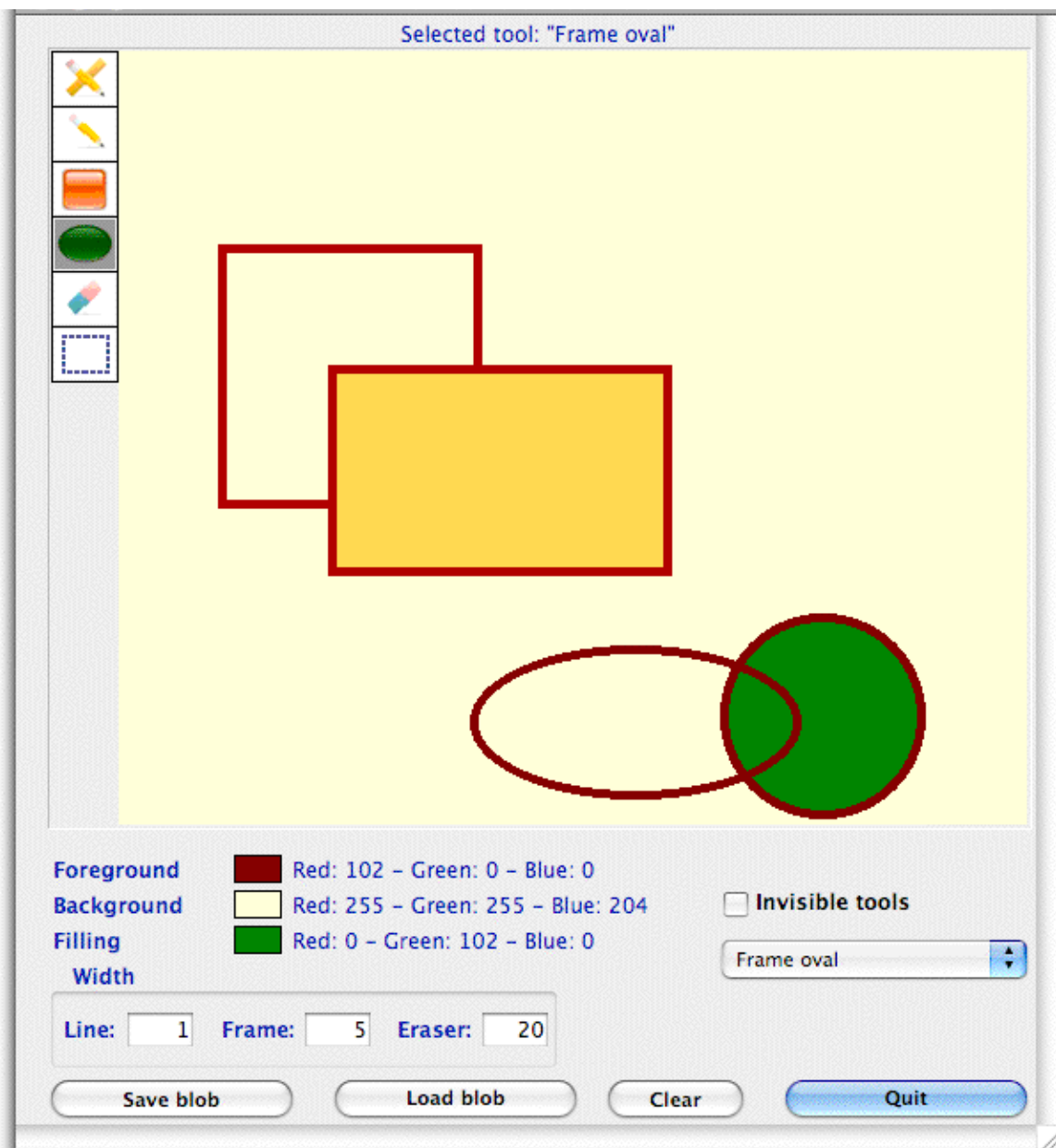
- 1 — Locate the 4th Dimension structure where you wish to install the 4D Line plug-in.
- 3 — If you don't already have a directory labeled "Plugins", create one now.
- 4 — Copy the following plug-in to your applications Plugins folder: 4DLine.bundle.

The 4D Line User Interface

The screenshot below is taken from the demonstration database.

Overview

This 4D interpreted database displays a 4D Line area with its built-in six tools. The bottom part contains the area's settings information — colors, widths, as well as controls allowing to modify the area behavior through 4D Line commands.



Tools

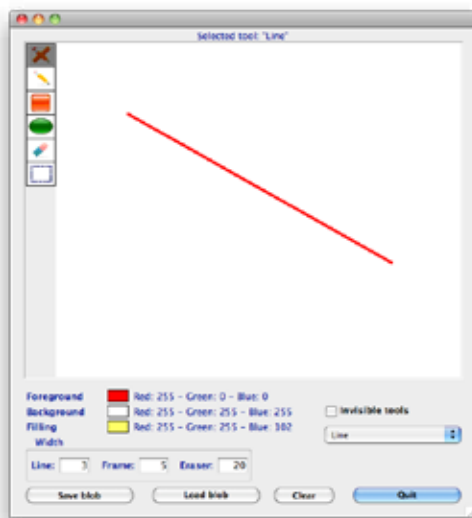


Line/Pen Tool

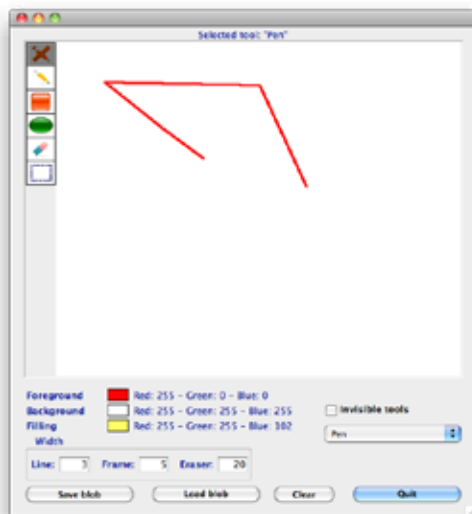
When this tool is selected, the user action in the 4D Line area will draw straight lines. The line width and color can be set with 4D Line commands. See the [Widths](#) and [Colors](#) sections.

This tool has two modes: **Line** and **Pen**.

- A single **click** on the tool sets the **Line mode**, where a straight line will be drawn from the point where the mouse is pressed down, to the point where the mouse is released:



- **Option/alt-click** on the tool sets the **Pen mode**, where the straight line will be drawn from the previous line's ending point, thus allowing continued lines:

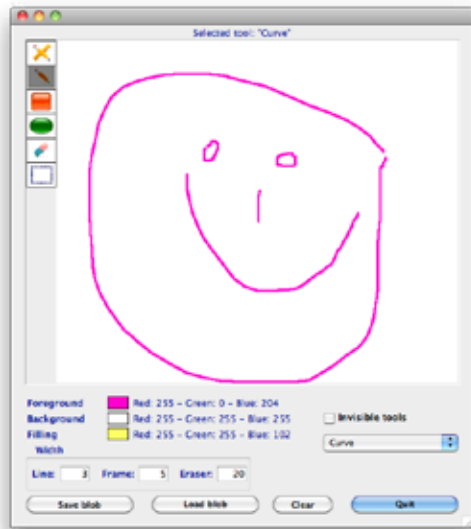


The 4D Line User Interface



Curve Tool

When this tool is selected, the user action in the 4D Line area will draw free lines, following the pointer while the mouse is down:



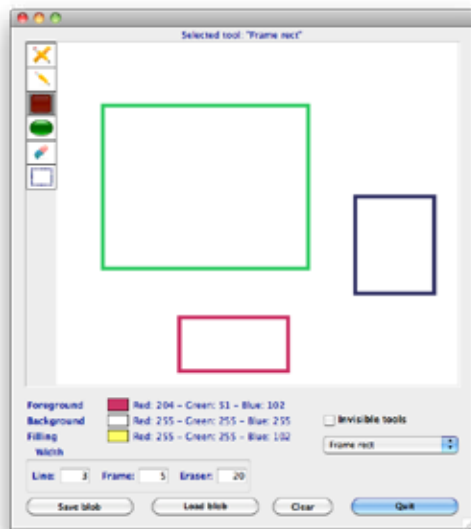
The line width and color can be set with 4D Line commands. See the [Widths](#) and [Colors](#) sections.



Rectangle Tool

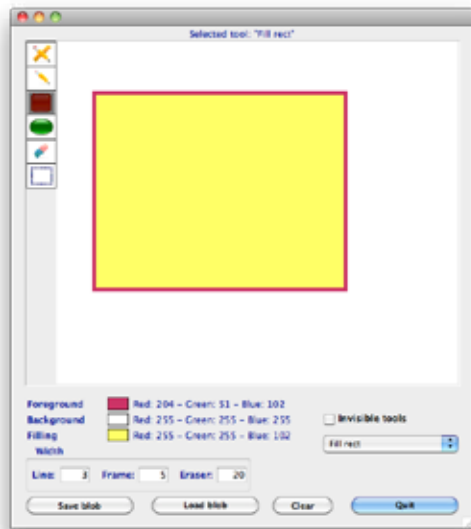
When this tool is selected, the user action in the 4D Line area will draw rectangles. The frame width and color, as well as the fill color can be set with 4D Line commands. See the [Widths](#) and [Colors](#) sections. This tool has two modes: **Frame rect** and **Fill rect**.

- A single **click** on the tool sets the **Frame rect mode**, where a rectangle will be drawn from the corner where the mouse is pressed down, to the opposite corner where the mouse is released:

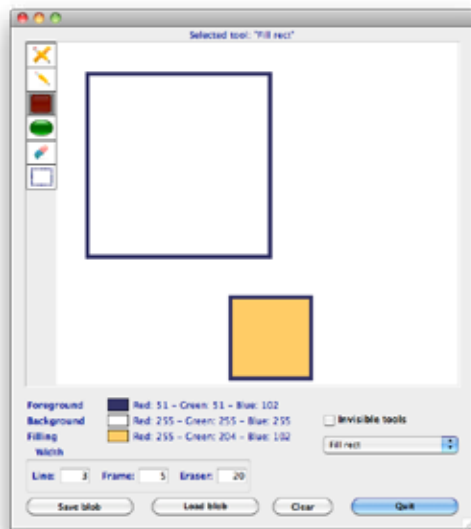


The 4D Line User Interface

- **Option/alt-click** on the tool sets the **Fill rect mode**, similar to the Frame rect mode but the rectangle will be filled with the current Fill color (see [Colors](#)):



- In addition, pressing the **Shift** key while drawing the rectangle will draw a **Square**. The square will be filled or not, depending on whether the Alt (option) key was pressed or not while selecting the [Rectangle Tool](#):



The 4D Line User Interface

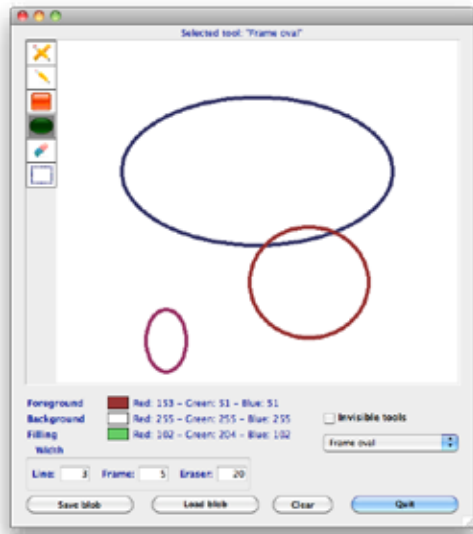


Oval Tool

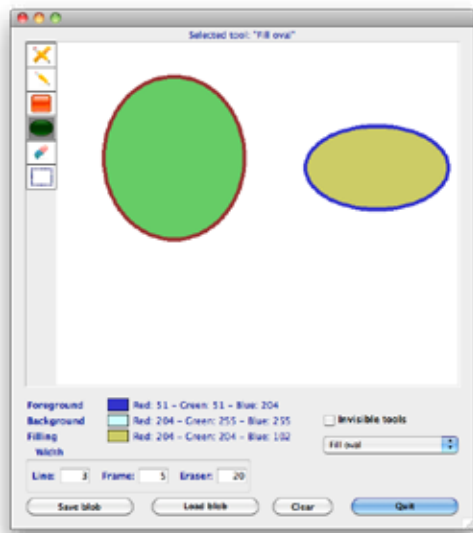
When this tool is selected, the user action in the 4D Line area will draw ovals.

This tool has two modes: **Frame oval** and **Fill oval**. The frame width and color, as well as the fill color can be set with 4D Line commands. See the [Widths](#) and [Colors](#) sections.

- A single **click** on the tool sets the **Frame oval mode**, where an oval will be drawn from the point where the mouse is pressed down, to the opposite point where the mouse is released:

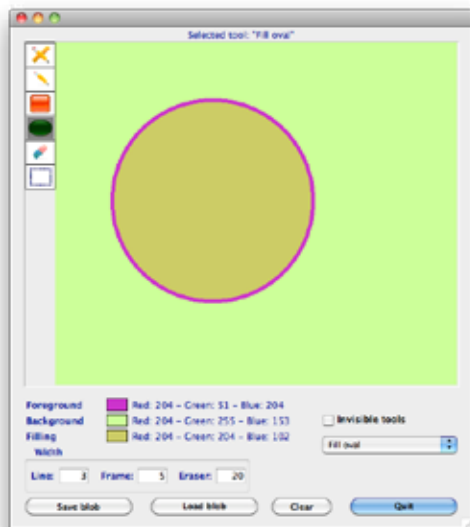


- **Option/alt-click** on the tool sets the **Fill oval mode**, similar to the Frame oval mode but the oval will be filled with the current Fill color (see [Colors](#)):



The 4D Line User Interface

- In addition, pressing the **Shift** key while drawing the oval will draw a **Circle**. The circle will be filled or not, depending on whether the Alt (option) key was pressed or not while selecting the [Oval Tool](#):



The 4D Line User Interface



Eraser Tool

Selecting the Eraser tool and pressing the mouse down colors the pixels below the pointer with the background color.

The Eraser is basically a brush, the default diameter being 3 pixels.

The eraser diameter and the background color can be set with 4D Line commands. See the [Widths](#) and [Colors](#) sections.



Selection Tool

The selection tool allows the user to create a rectangular selection within the picture displayed in the 4D Line area.

This selection can be cut, copied (and eventually pasted) or erased using the Edit Menu or the [Edit Menu](#) Commands.

Ctrl/command-click on any active (selected) tool deselects the tool.

Programming 4D Line

4D Line commands can be used to modify the 4D Line area and interface behavior **at any time**, or to help and/or emulate the user's action programmatically.

Widths

There are three types of widths:

- The **Line** width affects the [Line/Pen Tool](#) and the [Curve Tool](#).
- The **Frame** width affects the [Rectangle Tool](#) and the [Oval Tool](#).
- The **Eraser** width affects the [Eraser Tool](#): it is the “brush” diameter.

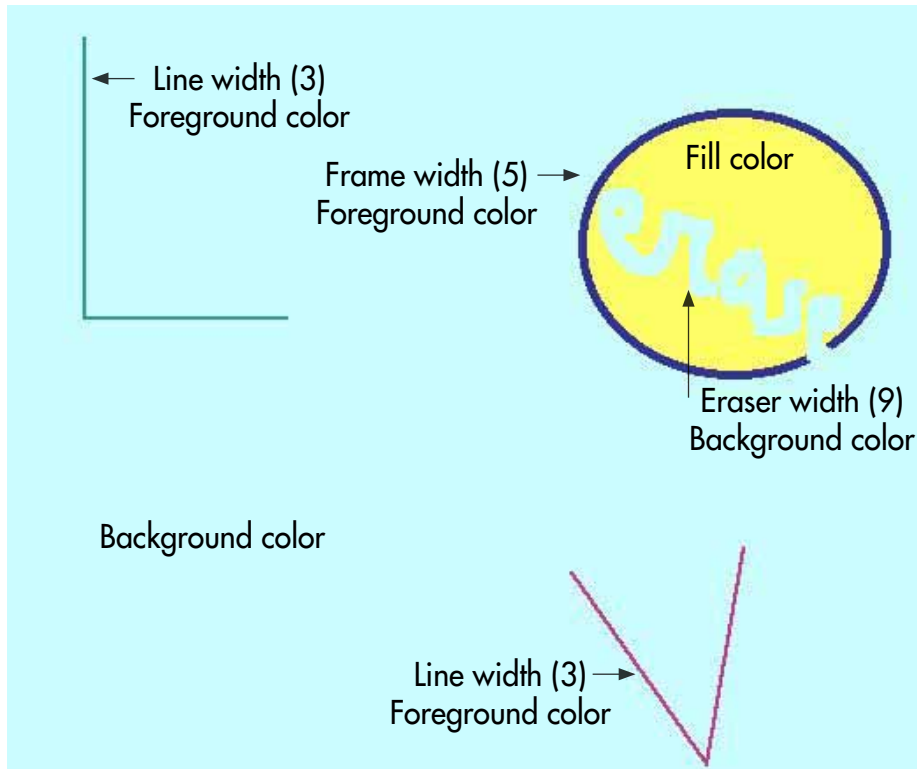
All three widths are set with a single command: [LN_SetWidths](#) and their current values are returned by [LN_GetWidths](#).

Colors

There are three types of colors:

- The **Foreground** color affects all four drawing tools: Line/Pen Tool, Curve Tool, Rectangle Tool and Oval Tool. It is set with the [LN_SetForegroundColor](#) command and its value is returned by [LN_GetForegroundColor](#).
- The **Background** color is the default area color (where nothing is drawn) and also affects the Eraser Tool. It is set with the [LN_SetBackgroundColor](#) command and its value is returned by [LN_GetBackgroundColor](#).
- The **Fill** color affects the Rectangle Tool and Oval Tool in Fill mode. It is set with the [LN_SetFillColor](#) command and its value is returned by [LN_GetFillColor](#).

Summary of Width and Color types



- The background color was set to light blue.
- The “L” sign was drawn with the Pen tool, a green foreground color and a 3 pixels line width.
- The “V” sign was drawn with the Pen tool, a magenta foreground color and a 3 pixels line width.
- The oval was drawn with the Oval tool in Fill oval mode, a dark blue frame color, a yellow fill color and a 5 pixels frame width.
- The eraser was used to write “erase” in the oval, with a 9 pixels width (diameter) and the background color still set to light blue.

Remember that all settings can be modified with 4D Line commands at any time, therefore the colors and widths may vary from an object to the other as you wish.

Edit Menu and Commands

Edit Menu Items and Commands Behavior

Sophisticated Edit Menu actions can be performed through 4D Line commands, or the Edit Menu itself.

- Undo/Redo is managed by [LN Undo](#) and [LN Redo](#). See [Multiple Undos/Redos](#) below.
- Cut is managed by [LN Cut](#).
- Copy is managed by [LN Copy](#).
- Paste is managed by [LN Paste](#).
- Erase is managed by [LN Delete](#).
- Select All is managed by [LN SelectAll](#).

In addition, the [Selection Tool](#) can be emulated with [LN SetSelect](#) and informations on the selection resulting of the user action on the tool (or [LN SetSelect](#) or [LN SelectAll](#)) can be read through [LN GetSelect](#).

The Menu and Commands behavior depends upon the existence of a current selection (user action or 4D Line commands):

Action	No current selection	Existing current selection
Cut	Dimmed No current selection error	Cuts current selection
Copy	Dimmed No current selection error	Copies current selection
Paste (if clipboard contains a bitmap picture)	Pastes from 0,0 (upper left corner) *	Pastes centered on the current selection center *
Paste (if clipboard empty or other contents)	Dimmed Clipboard empty error or Clipboard not a picture error	
Delete	Dimmed No current selection error	Deletes current selection
Select all	Selects all	

Green = menu items

Blue = commands

Red = menu items and commands

* When option/alt is held down while the Edit>Paste menu is selected, the clipboard picture is scaled into the destination rectangle (i.e. the whole area if no current selection, or the current selection).

Selection Tool and Specific Behaviors

In addition, specific behaviors depend upon the [Selection Tool](#) being active (selected) or not, and whether a current selection is in memory (resulting of the user action on the tool, or [LN_Select](#) or [LN_SelectAll](#)).

When a current selection is in memory, a selection rectangle is displayed only when the tool is active, but the selection is kept in memory and displayed again when the tool becomes active.

- If the selection tool is active and the current selection is not empty, both menu items and 4D Line commands behave according to the “Existing current selection” column on previous page.
- If the selection tool is active and the current selection is empty (no current selection), both menu items and 4D Line commands behave according to the “No current selection” column on previous page.
- If the selection tool is not active and the current selection is not empty, menu items are disabled and 4D Line commands behave according to the “Existing current selection” column on previous page.
- If the selection tool is not active and the current selection is empty (no current selection), menu items are disabled and 4D Line commands behave according to the “No current selection” column on previous page.

Multiple Undos/Redos

Undos and redos are available as a stack. You can roll back user actions in a row or cancel these undos (reinstate the actions), using either the Edit Menu or the [LN_Undo](#) and [LN_Redo](#) commands.

This undo/redo stack uses an internal memory buffer that can grow to a significant size (several Mb.)

Please contact our support if you heavily use this functionality, as we will be able to provide you with diagnostic tools to help trace your memory usage, modify the buffer size or the maximum undo count.

Loading and Saving

[LN_LoadPicture](#) and [LN_SavePicture](#) can be used to store and retrieve the picture displayed in the 4D Line area into/from a picture type 4D variable.

[LN_LoadBlob](#) and [LN_SaveBlob](#) can be used to store and retrieve the full 4D Line area along with its settings, **including its current selection if any** into/from a blob type 4D variable.

These 4D variables can be saved into 4D fields (picture or blob types) or external documents.

Configuration Commands

LN_Register

(registrationKey:S) → resultCode:L

Parameter	Type	Description
→ registrationKey	string	Registration key
← resultCode	longint	Result code (1 if successful)

LN_Register is used to register the 4D Line plug-in for standalone or server use.

You must call *LN_Register* with a valid registration key; otherwise 4D Line will operate in demonstration mode.

Without a valid registration key, 4D Line will operate in demonstration mode during 20 minutes.

Like all e-Node plug-ins, 4D Line offers six different license types. There are no such things as MacOS vs Windows or Development vs Deployment:

- **Single user license.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Standalone or Runtime. Since the registration key is linked to a specific 4D license, you need to provide the number returned by the 4D command **GET SERIAL INFORMATION** (first parameter). A new license will be provided for free at any time if you change your 4D version and/or get a new 4D registration key.
- **Small server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server up to 10 users. The registration key is linked to your 4D Server license just as above.
- **Medium server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server up with 11 to 20 users. The registration key is linked to your 4D Server license just as above.
- **Large server.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on 4D Server over 20 users. The registration key is linked to your 4D Server license just as above.
- **Unlimited Single User.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on as many 4D Standalone, Runtime or Engine copies that run your 4D application(s). This is a yearly license, which expires one month after the date when it is to be renewed. The expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire.**
- **Unlimited OEM.** This license allows development (interpreted mode) or deployment (interpreted or compiled mode) on as many 4D Server (of any number of users), 4D Standalone, Runtime or Engine copies that run your 4D application(s). This is a yearly license, which expires one month after the date when it is to be renewed. The expiration only affects interpreted mode. **Compiled applications using an obsolete license will never expire.**

Configuration Commands

A 4D database used to retrieve your 4D serial information is available from the following link:

<http://www.e-node.net/ftp/GetSerialInfo>

registrationKey — Pass the registration key to register your copy of 4D Line. Only one registration key is required. The key is either linked to the 4D or 4D Server serial number, or to the name of the company/developer.

resultCode — This will return a value of 1 if the registration key is valid and a value of 0 if the registration key is invalid. You should verify the correctness of the registration key by tracing over the call to **LN_Register** and examining **resultCode**.

Example:

```
C_LONGINT($result)
$result:=LN_Register("Place your registration key here")
If($result#1) `error
    ALERT("4D Line could not be registered:"+String($result))
End if
```

LN_SetWidths

(areaRef:L; lineWidth:L; outlineWidth:L; eraserWidth:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ lineWidth	longint	Line width in pixels
→ outlineWidth	longint	Outline width in pixels
→ eraserWidth	longint	Eraser width (diameter) in pixels
← errorCode	longint	Error code (or 0 if no error)

LN_SetWidths is used to set the widths for the three lines types.

Please read the [Widths](#) section for explanations about the three width types.

Default values are:

- 1 pixel for `lineWidth`.
- 1 pixel for `outlineWidth`.
- 3 pixels for `eraserWidth`.

Example:

```
`Set the widths to 2 pixels for lines, 3 for shape outlines and 10 for the eraser diameter  
$error:=LN_SetWidths(my4DlineArea;2;3;10)
```

LN_GetWidths

(areaRef:L; lineWidth:L; outlineWidth:L; eraserWidth:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← lineWidth	longint	Line width in pixels
← outlineWidth	longint	Outline width in pixels
← eraserWidth	longint	Eraser width (diameter) in pixels
← errorCode	longint	Error code (or 0 if no error)

LN_GetWidths is used to get the current width values for the three lines types into 4D variables.

Please read the [Widths](#) section for explanations about the three width types.

Example:

```
$error:=LN_SetWidths(my4DlineArea;varLine;varOutline;varEraser)
```

LN_SetForegroundColor

(areaRef:L; red:L; green:L; blue:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ red	longint	Foreground color red RGB component
→ green	longint	Foreground color green RGB component
→ blue	longint	Foreground color blue RGB component
← errorCode	longint	Error code (or 0 if no error)

LN_SetForegroundColor is used to set the foreground color through RGB values 0-255.

Please read the [Colors](#) section for explanations about the 4D Line color types.

0-255 RGB values can be extracted from a single 4D long integer using bitwise operations.

Example:

```
$redColor:=((foregroundColour >> 0x0010) & 0x00FF)
$greenColor:=((foregroundColour >> 0x0008) & 0x00FF)
$blueColor:=(foregroundColour & 0x00FF)
$error:=LN_SetForegroundColor(my4DlineArea;$redColor;$greenColor;$blueColor)
```

LN_GetForegroundColor

(areaRef:L; red:L; green:L; blue:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← red	longint	Foreground color red RGB component
← green	longint	Foreground color green RGB component
← blue	longint	Foreground color blue RGB component
← errorCode	longint	Error code (or 0 if no error)

LN_GetForegroundColor is used to get the current foreground color RGB values (0-255) into 4D variables.

Please read the [Colors](#) section for explanations about the 4D Line color types.

0-255 RGB values can be combined into a single 4D long integer using bitwise operations.

Configuration Commands

Example:

```
$error:=LN_GetForegroundColor (my4DlineArea;$redColor;$greenColor;$blueColor)
foregroundColour:= ($redColor << 16)+($greenColor << 8)+$blueColor
foregroundInformation:="Red: "+String($redColor)+" - "+"Green: "+String($greenColor)+" - "
+"Blue: "+String($blueColor)
```

LN_SetBackgroundColor

(areaRef:L; red:L; green:L; blue:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ red	longint	Background color red RGB component
→ green	longint	Background color green RGB component
→ blue	longint	Background color blue RGB component
← errorCode	longint	Error code (or 0 if no error)

LN_SetBackgroundColor is used to set the background color through RGB values 0-255.

Please read the [Colors](#) section for explanations about the 4D Line color types.

0-255 RGB values can be extracted from a single 4D long integer using bitwise operations.

Refer to the example provided for the [LN_SetForegroundColor](#) command.

LN_GetBackgroundColor

(areaRef:L; red:L; green:L; blue:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← red	longint	Background color red RGB component
← green	longint	Background color green RGB component
← blue	longint	Background color blue RGB component
← errorCode	longint	Error code (or 0 if no error)

LN_GetBackgroundColor is used to get the current background color RGB values (0-255) into 4D variables.

Please read the [Colors](#) section for explanations about the 4D Line color types.

0-255 RGB values can be combined into a single 4D long integer using bitwise operations.

Refer to the example provided for the [LN_GetForegroundColor](#) command.

4D Line Commands Syntax and Examples

LN_SetFillColor

(areaRef:L; red:L; green:L; blue:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ red	longint	Fill color red RGB component
→ green	longint	Fill color green RGB component
→ blue	longint	Fill color blue RGB component
← errorCode	longint	Error code (or 0 if no error)

LN_SetFillColor is used to set the fill color through RGB values 0-255.

Please read the [Colors](#) section for explanations about the 4D Line color types.

0-255 RGB values can be extracted from a single 4D long integer using bitwise operations.

Refer to the example provided for the [LN_SetForegroundColor](#) command.

LN_GetFillColor

(areaRef:L; red:L; green:L; blue:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← red	longint	Fill color red RGB component
← green	longint	Fill color green RGB component
← blue	longint	Fill color blue RGB component
← errorCode	longint	Error code (or 0 if no error)

LN_GetFillColor is used to get the current fill color RGB values (0-255) into 4D variables.

Please read the [Colors](#) section for explanations about the 4D Line color types.

0-255 RGB values can be combined into a single 4D long integer using bitwise operations.

Refer to the example provided for the [LN_GetForegroundColor](#) command.

Saving and Loading Commands

LN_LoadPicture

(areaRef:L; picture:P) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ picture	picture	4D picture type variable
← errorCode	longint	Error code (or 0 if no error)

LN_LoadPicture loads a 4D picture and displays it in the 4D Line area for editing.

Please read the [Loading and Saving](#) section for details about saving and loading 4D Line objects.

Example:

```
READ PICTURE FILE("";$picture) `read from an external document
If (OK=1)
  $error:=LN_LoadPicture (my4DlineArea;$picture) `display the picture contained in the variable
End if
```

LN_SavePicture

(areaRef:L; picture:P) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ picture	picture	4D picture type variable
← errorCode	longint	Error code (or 0 if no error)

LN_SavePicture saves the 4D Line area contents as a standard 4D picture variable.

Please read the [Loading and Saving](#) section for details about saving and loading 4D Line objects.

Example:

```
$error:=LN_SavePicture (my4DlineArea;$picture) `save the picture displayed in the area into a variable
If ($error=0)
  WRITE PICTURE FILE("";$picture) `save the variable to an external document
End if
```

Saving and Loading Commands

LN_LoadBlob

(areaRef:L; blob:X) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ blob	blob	4D blob type variable
← errorCode	longint	Error code (or 0 if no error)

LN_LoadBlob loads a 4D blob previously saved with [LN_SaveBlob](#) and displays the saved 4D Line area for editing, including its picture, current selection and settings.

Please read the [Loading and Saving](#) section for details about saving and loading 4D Line objects.

Example:

```
$error:=LN_LoadBlob (my4DlineArea;[Dialogs]BlobImage) `read directly from a 4D blob field
```

LN_SaveBlob

(areaRef:L; picture:P) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ picture	picture	4D picture type variable
← errorCode	longint	Error code (or 0 if no error)

LN_SaveBlob saves the 4D Line area, including its picture, current selection and settings into a 4D blob variable, which can eventually be loaded with [LN_LoadBlob](#) for future editing.

Please read the [Loading and Saving](#) section for details about saving and loading 4D Line objects.

Example:

```
$error:=LN_SaveBlob (my4DlineArea;$blob) `do NOT save directly into a field — a variable is required  
If ($error=0)  
  [Dialogs]BlobImage:=$blob `save to the 4D field  
  SAVE RECORD([Dialogs])  
End if
```

Tools Commands

LN_SetToolBarProperties

(areaRef:L; selectedTool:L; hidden;L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ selectedTool	longint	Tool to select (or 0 to deselect)
→ hidden	longint	0 to display the toolbar, 1 to hide it
← errorCode	longint	Error code (or 0 if no error)

LN_SetToolBarProperties defines the tool to select if any, and the toolbar visible status.

selectedTool — Deselects the currently selected tool (0) or selects (highlights) the specified tool (1 to 9):

- -1 - No change (the currently selected tool status is unchanged)
- 0 - Deselects the currently selected tool if any (default)
- 1 - [Line](#)
- 2 - [Curve](#)
- 3 - [Fill rect](#)
- 4 - [Fill oval](#)
- 5 - [Eraser](#)
- 6 - [Selection](#)
- 7 - [Pen](#)
- 8 - [Frame rect](#)
- 9 - [Frame oval](#)

hidden — Hides or shows the 4D Line area toolbar:

- If the value is -1, no change (the toolbar visibility status is unchanged)
- If the value is 0, displays the toolbar (default)
- If the value is 1, hides the toolbar: the entire 4D Line area can be used for drawing

Please read the [Tools](#) section for a presentation of 4D Line tools.

Example:

```
$error:=LN_SetToolBarProperties (my4DlineArea;5;0) `select the eraser tool, display the toolbar
```

LN_GetToolBarProperties

(areaRef:L; selectedTool:L; hidden;L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← selectedTool	longint	Selected tool (0 = no selected tool)
← hidden	longint	0 if the toolbar is visible, 1 if hidden
← errorCode	longint	Error code (or 0 if no error)

LN_GetToolBarProperties returns the 4D Line tool ID of the currently selected tool, or 0 if no tool is currently selected, as a result of a Ctrl/command-click on the active (selected) tool, or a call to [LN_SetToolBarProperties](#) with 0 as the second parameter.

See the [LN_SetToolBarProperties](#) command for a list of all available tool IDs.

This command also returns the visible status of the toolbar, as defined by [LN_SetToolBarProperties](#):

- If the value is 0, the toolbar is displayed (default)
- If the value is 1, the toolbar is hidden: the entire 4D Line area can be used for drawing

Please read the [Tools](#) section for a presentation of 4D Line tools.

Example:

```
$error:=LN_GetToolBarProperties (my4DlineArea;$tool;$hidden)
```

Selection and Edit Commands

LN_SetSelect

(areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ left	longint	Left coordinate
→ top	longint	Top coordinate
→ right	longint	Right coordinate
→ bottom	longint	Bottom coordinate
← errorCode	longint	Error code (or 0 if no error)

LN_SetSelect sets the selection (dotted frame indicating which part to cut, copy, delete or paste into) according to the four coordinates in pixels. The effect is similar to the user action on the [Selection Tool](#).

If all four coordinates left, top, right and bottom are set to 0, the current selection (if any) is deselected.

Please read the [Edit Menu and Commands](#) section for details about the edit menu and associated commands regarding the current selection concept in 4D Line.

LN_GetSelect

(areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← left	longint	Left coordinate
← top	longint	Top coordinate
← right	longint	Right coordinate
← bottom	longint	Bottom coordinate
← errorCode	longint	Error code (or 0 if no error)

LN_GetSelect returns the four coordinates in pixels of the current selection, whether resulting of the user action on the [Selection Tool](#) or a call to [LN_SetSelect](#).

If there is no current selection, all four coordinates contain the 0 value.

Selection and Edit Commands

Example:

```
$error:=LN_GetSelect (my4DlineArea;$left;$top;$right;$bottom)
If ($error=0)
  ALERT("Upper left: "+String($top)+"/"+String($left)+Char(Carriage return)+"Lower right: "
    +String($bottom)+"/"+String($right))
End if
```

LN_Cut

(areaRef:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← errorCode	longint	Error code (or 0 if no error)

LN_Cut cuts the current selection and copies its content into the clipboard.

If there is no current selection to cut, a No current selection error (number -101) is returned.

Please read the [Edit Menu and Commands](#) section for details.

Example:

```
$error:=LN_Cut (my4DlineArea)
```

LN_Copy

(areaRef:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← errorCode	longint	Error code (or 0 if no error)

LN_Copy copies the current selection (if any) into the clipboard.

If there is no current selection to copy, a No current selection error (number -101) is returned.

Please read the [Edit Menu and Commands](#) section for details.

Selection and Edit Commands

LN_Paste

(areaRef:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← errorCode	longint	Error code (or 0 if no error)

LN_Paste pastes the clipboard content (if applicable).

- If there is no current selection, the clipboard content is pasted from the 4D Line area upper left corner at coordinates 1,1,1,1.
- If there is a current selection, the clipboard content is pasted centered on the current selection center.

If the clipboard is empty or contains no valid (picture) information, a Clipboard empty error (number -100 or a Clipboard not a picture error (number -102) is returned.

Please read the [Edit Menu and Commands](#) section for details.

LN_Delete

(areaRef:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← errorCode	longint	Error code (or 0 if no error)

LN_Delete deletes the current selection without copying it to the clipboard. The selection is replaced by the current background color.

If there is no current selection to delete, a No current selection error (number -101) is returned.

Please read the [Edit Menu and Commands](#) section for details.

Selection and Edit Commands

LN_SelectAll

(areaRef:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← errorCode	longint	Error code (or 0 if no error)

LN_SelectAll selects the entire 4D Line area.

Please read the [Edit Menu and Commands](#) section for details.

LN_ClearArea

(areaRef:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← errorCode	longint	Error code (or 0 if no error)

LN_ClearArea clears (deletes) the entire 4D Line area. The area is filled with the current background color.

This command is equivalent to a [LN_SelectAll](#) followed by [LN_Delete](#), except that the current selection rectangle is preserved (if any).

Selection and Edit Commands

LN_Undo

(areaRef:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← errorCode	longint	Error code (or 0 if no error)

LN_Undo cancels the last action performed on the 4D Line area, whether by the user or a 4D Line command.

Multiple Undos/Redos are available. Please read the [Multiple Undos/Redos](#) section for details.

If there is nothing to undo or if undo is not possible, a Undo/Redo not possible error (number -4) is returned.

Example:

```
$error:=LN_Undo (my4DlineArea)
```

LN_Redo

(areaRef:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
← errorCode	longint	Error code (or 0 if no error)

LN_Redo reinstates the last action that was undone by [LN_Undo](#).

Multiple Undos/Redos are available. Please read the [Multiple Undos/Redos](#) section for details.

If there is nothing to redo or if redo is not possible, a Undo/Redo not possible error (number -4) is returned.

Drawing Commands

LN_DrawLine

(areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ left	longint	Left coordinate
→ top	longint	Top coordinate
→ right	longint	Right coordinate
→ bottom	longint	Bottom coordinate
← errorCode	longint	Error code (or 0 if no error)

LN_DrawLine draws a line from the **left/top** coordinates to the **right/bottom** coordinates, with the currently set [foreground color](#) and [line width](#).

This command emulates an action on the [Line/Pen Tool](#).

Example:

```
$error:=LN_DrawLine (my4DlineArea;$left;$top;$right;$bottom)
```

LN_DrawFrameRect

(areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ left	longint	Left coordinate
→ top	longint	Top coordinate
→ right	longint	Right coordinate
→ bottom	longint	Bottom coordinate
← errorCode	longint	Error code (or 0 if no error)

LN_DrawFrameRect draws a [frame rect](#) from the **left/top** coordinates to the **right/bottom** coordinates, with the currently set [foreground color](#) and [frame width](#).

This command emulates an action on the [Rectangle Tool](#).

LN_DrawFillRect

(areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ left	longint	Left coordinate
→ top	longint	Top coordinate
→ right	longint	Right coordinate
→ bottom	longint	Bottom coordinate
← errorCode	longint	Error code (or 0 if no error)

LN_DrawFillRect draws a [fill rect](#) from the **left/top** coordinates to the **right/bottom** coordinates, with the currently set [foreground color](#), [background color](#) and [frame width](#).

This command emulates an action on the [Rectangle Tool](#) while the **option/alt** key is pressed.

LN_DrawFrameOval

(areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ left	longint	Left coordinate
→ top	longint	Top coordinate
→ right	longint	Right coordinate
→ bottom	longint	Bottom coordinate
← errorCode	longint	Error code (or 0 if no error)

LN_DrawFrameOval draws a [frame oval](#) from the **left/top** coordinates to the **right/bottom** coordinates, with the currently set [foreground color](#) and [frame width](#).

This command emulates an action on the [Oval Tool](#).

LN_DrawFillOval

(areaRef:L; left:L; top:L; right:L; bottom:L) → errorCode:L

Parameter	Type	Description
→ areaRef	longint	Reference of 4D Line object on layout
→ left	longint	Left coordinate
→ top	longint	Top coordinate
→ right	longint	Right coordinate
→ bottom	longint	Bottom coordinate
← errorCode	longint	Error code (or 0 if no error)

LN_DrawFillOval draws a [fill oval](#) from the **left/top** coordinates to the **right/bottom** coordinates, with the currently set [foreground color](#), [background color](#) and [frame width](#).

This command emulates an action on the [Oval Tool](#) while the **option/alt** key is pressed.

4D Line Error Codes

Error	Value
Invalid area reference	-1
Expired demonstration	-2
Invalid blob	-3
Undo/Redo not possible	-4
Parameter error	-50
Clipboard empty	-100
No current selection	-101
Clipboard not a picture	-102
Memory full	-108