

Linux Desktop Testing Project - LDTP



Linux Desktop Testing Project – LDTP

<http://ldtp.freedesktop.org>

Tutorial

Linux Desktop Testing Project - LDTP

Copyright 2004 - 2007 Novell, Inc.
Copyright 2008-10 Nagappan Alagappan

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Lesser General Public License, Version 2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Lesser General Public License".

You should have received a copy of the GNU GNU Lesser General Public License along with this documentation; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Author

Nagappan A <nagappan@gmail.com>

Contributors:

Harsha <nharsha@gmail.com>

Premkumar J <prem.jothimani@gmail.com>

Guofu Xu <lavixu@gmail.com>

Surendran M <suren.silverprince@gmail.com>

Vamsi B <vamsi1985@gmail.com>

Table of Contents

About LDTP.....	5
Audience.....	5
About testing.....	5
Why testing ?.....	5
Why automation ?.....	5
Complexity of GUI testing ?.....	5
What type of testing can be done using LDTP ?.....	6
Advantage of accessibility based testing.....	6
Disadvantage of accessibility based testing.....	6
What applications can be tested ?.....	6
Supported platforms.....	6
LDTP Features.....	6
Web / Contact.....	7
LDTP on web.....	7
Development mailing list.....	7
Internet relay chatting – IRC.....	7
System requirements.....	7
Disk space requirement.....	7
Software requirements.....	7
Install the following dependency packages.....	7
Optional packages.....	7
Environment.....	7
Setup LDTP.....	7
Download source from GIT.....	7
Download source in tar ball format.....	8
Setup LDTP from source in Linux environment.....	8
Setup LDTP from binary.....	8
Architecture.....	8
LDTP Overall Architecture.....	8
LDTP Internals.....	8
Server.....	9
Client Handler.....	9
Component Handler.....	9
Event Handler.....	9
Logger.....	10
LDTP conventions.....	10
Appmap.....	10
Appmap convention.....	10
How to Access UI Objects from LDTP scripts.....	12
Window name.....	12
Different window types.....	12
Glob pattern support.....	13
Different ways of representing window name.....	13

Linux Desktop Testing Project - LDTP

Window name formats.....	13
Example.....	13
Object name.....	13
Label.....	13
Example.....	13
Label by / Label for (associated label).....	14
Example.....	14
Object name with out label / associated label accessing via index.....	14
Example.....	14
Object name with index.....	14
Example.....	14
Object name formats.....	14
Example.....	14
Accessibility library.....	14
Enabling accessibility.....	15
Importing LDTP modules.....	15
Example 1:	15
Example 2:.....	16
Call a function to perform an operation.....	16
LDTP API.....	17
Using / Hacking LDTP.....	17
Identifying controls.....	17
Push button.....	17
Menu item.....	18
Toggle button.....	19
Text control.....	19
Table control.....	20
Push button.....	21
Check box.....	21
Spin button.....	23
Page tab.....	24
Check menu item.....	24
Radio menu item.....	26
Combo box – Menu item.....	26
Combo box – List item.....	27
Launch application.....	28
GUI exist.....	29
Timeout.....	29
Generate raw keyboard events.....	30
Generate raw mouse events.....	31
Application information.....	31
Callback – On new window creation.....	32
Advantage.....	32
Example.....	32
Logging.....	33
Example script.....	33

Linux Desktop Testing Project - LDTP

How to execute LDTP scripts.....	33
Suggestions from LDTP team.....	34
How to operate LDTP from a remote system.....	34
LDTP engine.....	34
LDTP client.....	34
Troubleshooting LDTP.....	34
Bibliography.....	36

About LDTP

Linux Desktop Testing Project (LDTP) is aimed at producing high quality test automation framework and cutting-edge tools that can be used to test GNU/Linux Desktop and improve it. It uses the **Accessibility** libraries to poke through the application's user interface. The framework also has tools to record test-cases based on user-selection on the application.

LDTP core framework uses *Appmap* (application map) and the recorded *test-cases* to test an application and gives the status of each test-case as output. As of now, LDTP can test any GNOME application which are accessibility enabled, [Mozilla](#), Openoffice.org, any [Java](#) application (should have a UI based on swing) and KDE 4.0 applications based on QT 4.0 (based on the press releases by [KDE](#)).

We encourage you to join the project and help us to create robust, reliable and stable test tool / framework for Unix Desktops. Thanks to **GNOME** Accessibility team and **Sun Microsystems** Accessibility team for their great work and their continuous support !!!

Audience

Its assumed that the user of this document has little knowledge about UI controls in any GUI application, minimal Linux or Unix (Solaris / BSD flavor) knowledge.

About testing

Why testing ?

Testing is a process to identify defects in a (software) system. For more information - http://en.wikipedia.org/wiki/Software_testing

Why automation ?

Testing an application multiple times with same steps, automated process can do a better job.

Complexity of GUI testing ?

- Identification of object in a window (push button, menu item)
- Should be context sensitive (Window specific operation)
- Handling of unexpected pop-up windows
- Keeping the test script in sync with the UI changes
- Failures has to be verified on each operation
- Rendering of images / text in the display area

What type of testing can be done using LDTP ?

LDTP can be used to test the functionality of any accessibility enabled application.

Advantage of accessibility based testing

- Accessibility libraries provide applications property, state, its child items etc.
- No need to work in toolkit (GTK, AWT, QT) level

Disadvantage of accessibility based testing

- Application which are not accessibility enabled can't be tested

What applications can be tested ?

As of now, LDTP can test any GNOME application which are accessibility enabled, [Mozilla](#), Openoffice.org, any [Java](#) application (should have a UI based on swing) and KDE 4.0 applications based on QT 4.0 (based on the press releases by [KDE](#)).

Supported platforms

- OpenSuSE
- OpenSolaris
- Debian
- Ubuntu
- Fedora
- FreeBSD
- Embedded Platform (Palm Source / Access Company)

LDTP Features

- LDTP concepts are derived from [Software Automation Framework Support](#)
- LDTP supports verification of actions performed (guiexist, verifystate, etc) - [API Reference](#)
- Writing test scripts are very easy, the script writer need not know about the object hierarchy
- CPU / Memory performance monitoring of application-under-test can be measured - [Class pstats](#)
- From the XML log, we can gather [HTML report](#) using XSLT
- Group based execution, which provides precise control on the flow of test-script execution - [ldtprunner XML](#)
- Scripts can be written as a reusable component and for that the data can be stored / retrieved in XML - [Data XML](#)
- Objects are identified both [statically](#) and [dynamically](#)

Web / Contact

LDTP on web

- Website - <http://ldtp.freedesktop.org>
- Source / Binary - <http://ldtp.freedesktop.org/wiki/Download>
- API reference - <http://ldtp.freedesktop.org/wiki/Docs>
- HOWTO - <http://ldtp.freedesktop.org/wiki/HOWTO>
- FAQ - <http://ldtp.freedesktop.org/wiki/FAQ>

Development mailing list

<http://lists.freedesktop.org/mailman/listinfo/ldtp-dev>

Internet relay chatting – IRC

#ldtp of irc.freenod.net

System requirements

Disk space requirement

Less than 450 KB

Software requirements

Install the following dependency packages

- python-atspi or relevant name in your distribution
- twisted-web or relevant name in your distribution
- python-gnome or relevant name in your distribution

Optional packages

- [Import](#) tool of [ImageMagick](#) - To capture a screenshot
- [Python Imaging Library](#) - Compare images, black out a region in an image
- [pystatgrab](#) - CPU / Memory utilization monitoring library

Environment

GNOME 2.16 or above / **KDE 4.0** or above

Setup LDTP

Download source from GIT

- check out source from [GIT](#) with the following command: 'git clone git://anongit.freedesktop.org/git/ldtp/ldtp2.git'.

Linux Desktop Testing Project - LDTP

- change to the source directory with the following command: 'cd ldtp'

Download source in tar ball format

- If you can't access GIT, get the source tarball from <http://ldtp.freedesktop.org/wiki/Download>
- Extract and change to the source directory with the following command: 'cd ldtp-<version number>'

Setup LDTP from source in Linux environment

- build with 'python setup.py build'
- setup with 'sudo python setup.py install' as the root user

Setup LDTP from binary

Download latest RPM / Deb / Gentoo / Solaris package from <http://ldtp.freedesktop.org/wiki/Download>

Architecture

LDTP Overall Architecture

Test scripts uses LDTP API interface, which in-turn communicate to LDTP engine either by UNIX socket or by TCP socket. LDTP engine talks to Application under test (AUT) using AT-SPI library.



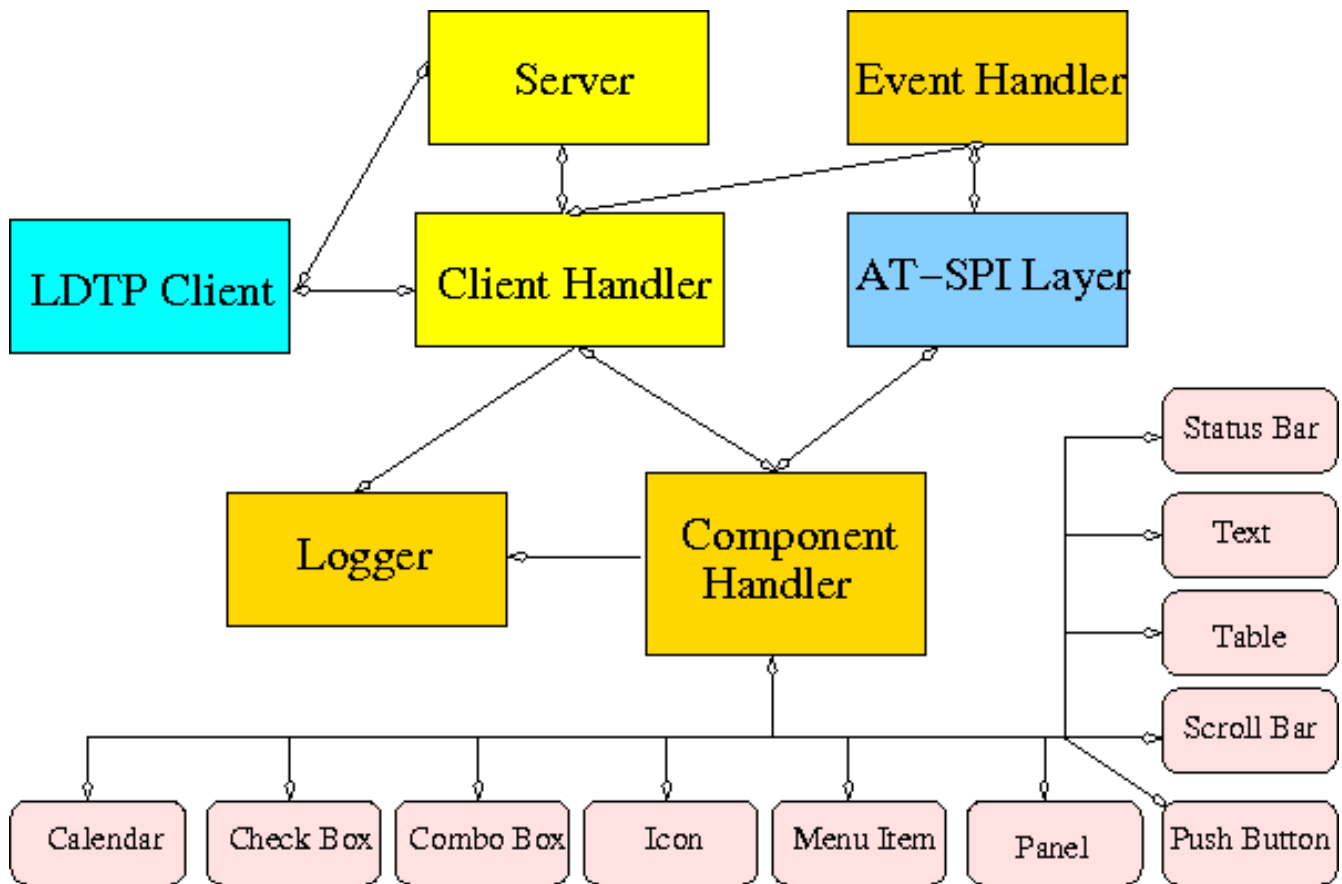
LDTP Internals

LDTP Clients can talk to LDTP engine with the set of LDTP Command Transfer Protocol (LDTPCTP) as defined in the Docs section. LDTP client wraps the LDTP Command Transfer Protocol based on the programming language (example: separate python client, maybe in future we can have Mono / Java / Perl based clients). Communication between the client and server takes place in XML format.

Most of LDTP ideas are implemented from <http://safesdev.sf.net> Most of the commands takes at-least 2 arguments. First argument will be context (window in which we want to operate) and the second argument will be component (object in which we want to operate, based on the current context).

Example: click ('*-gedit', 'btnNew') # Click operation will be performed on a window which is having *-gedit (regexp) and in that window object name 'New', which is of type 'push button'.

Linux Desktop Testing Project - LDTP



Server

When a test script is started, the LDTP client will establish a connection with the LDTP engine using AF_UNIX / AF_INET.

Client Handler

When ever a command is executed from the script, the client frames the XML data and send it to the server. LDTP engine parses the command request from the client and invoke the respective Component Handler.

Component Handler

Each individual component handlers uses the AT-SPI libraries to communicate to the respective application. Based on the execution status, success or failure will be notified as a response (in XML format) to the client. In few cases the requested data from the respective component will be returned to the client, based on the request (example: gettextvalue).

Event Handler

For unexpected windows (example: connection reset by peer /connection timed out dialogs) can be handled by registering a callback function and the respective callback function will be called, whenever

Linux Desktop Testing Project - LDTP

the registered window with the title appears and even this window could be based on regular expression.

Logger

Logs the execution status in XML format

LDTP conventions

Appmap

'Appmap' [Application Map in short] is a text based representation of the GUI which is under testing. Each and every UI control viz., Button, Text Box etc., are represented using predefined conventions (which are listed in the table below) along with their parent UI object information. At runtime, a particular UI control is accessed by using the Appmap generated for the GUI under testing. For more details about Appmap refer

<http://safesdev.sourceforge.net/DataDrivenTestAutomationFrameworks.htm#TheApplicationMap>

Appmap convention

Class keywords	convention used in appmap
ACCEL_LABEL	
ALERT	dlg
ANIMATION	
ARROW	
CALENDAR	cal
CANVAS	cnvs
CHECK_BOX	chk
CHECK_MENU_ITEM	mnu
COLOR_CHOOSER	
COLUMN_HEADER	
COMBO_BOX	cbo
DATE_EDITOR	
DESKTOP_ICON	
DESKTOP_FRAME	frm
DIAL	dial
DIALOG	dlg
DIRECTORY_PANE	
DRAWING_AREA	dwg
FILE_CHOOSER	dlg

Linux Desktop Testing Project - LDTP

FILLER	flr
FONT_CHOOSER	dlg
FRAME	frm
GLASS_PANE	
HTML_CONTAINER	html
ICON	ico
IMAGE	img
INTERNAL_FRAME	
LABEL	lbl
LAYERED_PANE	pane
LIST	lst
LIST_ITEM	lsti
MENU	mnu
MENU_BAR	mbar
MENU_ITEM	mnu
OPTION_PANE	opan
PAGE_TAB	ptab
PAGE_TAB_LIST	ptl
PANEL	pnl
PASSWORD_TEXT	txt
POPUP_MENU	pop
PROGRESS_BAR	pbar
PUSH_BUTTON	btn
RADIO_BUTTON	rbtn
RADIO_MENU_ITEM	mnu
ROOT_PANE	rpan
ROW_HEADER	rhdr
SCROLL_BAR	scbr
SCROLL_PANE	scpn
SEPARATOR	sep
SLIDER	sldr
SPIN_BUTTON	sbtn
SPLIT_PANE	splt
STATUS_BAR	stat
TABLE	tbl

Linux Desktop Testing Project - LDTP

TABLE_CELL	tbl
TABLE_COLUMN_HEADER	tch
TABLE_ROW_HEADER	trh
TEAROFF_MENU_ITEM	tmi
TERMINAL	term
TEXT	txt
TOGGLE_BUTTON	tbtn
TOOL_BAR	tbar
TOOL_TIP	ttip
TREE	tree
TREE_TABLE	ttbl
UNKNOWN	unk
VIEWPORT	view
WINDOW	dlg
EXTENDED	
HEADER	hdr
FOOTER	foot
PARAGRAPH	para
RULER	rul
APPLICATION	app
AUTOCOMPLETE	txt
CALENDARVIEW	cal
CALENDAREVENT	cal
EDITBAR	txt
ENTRY	txt

How to Access UI Objects from LDTP scripts

Two main entities to act on an object, window name, object name.

Window name

To operate on a window, we need to know the window name (nothing but window title).

Different window types

1. Frame (frm)
2. Dialog (dlg)
3. Alert (dlg)

Linux Desktop Testing Project - LDTP

4. Font Chooser (dlg)
5. File Chooser (dlg)
6. Window (This type in general does not have any associated title, so we need to represent them using index - dlg)

Glob pattern support

Window name can be clubbed with glob patterns (* or ?)

Different ways of representing window name

1. Window type and window title (Ex: 'frmUnsavedDocument1-gedit')
2. Window title (Ex: 'Unsaved Document 1 - gedit')
3. Window type, glob expression and partial window title (Ex: 'frm*-gedit')
4. Glob pattern and partial window title (Ex: '*-gedit')
5. Window type, partial window title and glob pattern (Ex: 'frmUnsavedDocument1*')
6. Window type, window title and index (If two windows of same title exist at same time. Ex: First window name 'dlgAppointment', Second window name 'dlgAppointment1')
7. Window type and index (only if window does not have any accessible title, Ex: 'dlg0')

Window name formats

If window label contains space or new line characters, they will be stripped.

Example

1. 'Unsaved Document 1 – gedit', will be represented as 'UnsavedDocument1-gedit'
2. 'Unsaved Document 1
-
gedit', will be represented as 'UnsavedDocument1-gedit'

Object name

Object (the type of control in which we want to operate) can be identified either with a label or by an associated label.

Label

In general *menu / menu item / push button / toggle button* type controls can be accessed through its label.

Example

mnuFile (gedit menu)

mnuNew (gedit menu item)

btnNew (gedit tool bar, push button)

tbtnLocation (gedit Open File dialog, toggle bar control)

Label by / Label for (associated label)

In general *text / tables / check box / radio button / spin button / combo box* controls can be accessed using the associated label only.

Example

txtLocation (gedit Open File dialog, text control)
tblFiles (gedit Open File dialog, table control)
cboSearchfor (gedit Find dialog, combo box control)
chkMatchcase (gedit Find dialog, check box control)
sbtnRightmarginatcolumn (gedit Preferences dialog, spin button control)

Object name with out label / associated label accessing via index

If a control does not have any label or associated label, then it can be accessed using index.

Example

txt0 (gedit text rendering region)
ptl0 (gedit Preferences dialog, page tab list control)
ptl0 (In gedit when more than one files are opened, a page tab list control will be available)

Object name with index

In some cases, a control type can be present in multiple places in the same window and chances that it may have same label too in that case, the first control can be accessed just with the default notation, but the second control and further can be accessed with the format control type, label or associated label and index starting from 1.

Example

btnAdd – First push button control with label Add
btnAdd1 – Second push button control with label Add
btnAdd2 – Third push button control with label Add

Object name formats

If object label or associated label contains space, dot, colon, under score or new line characters, they will be stripped.

Example

'Search for:' will be represented as 'Searchfor'

'File name 'a_txt' already exist.

'Replace' will be represented as 'Filename'atxt'alreadyexistReplace'.

Accessibility library

LDTP uses accessibility libraries (at-spi) available in GNOME environment. Using accessibility we can get the information about application and its current state (property). We can be able to poke

Linux Desktop Testing Project - LDTP

through each layer in any application, if and only if, the application is accessibility enabled.

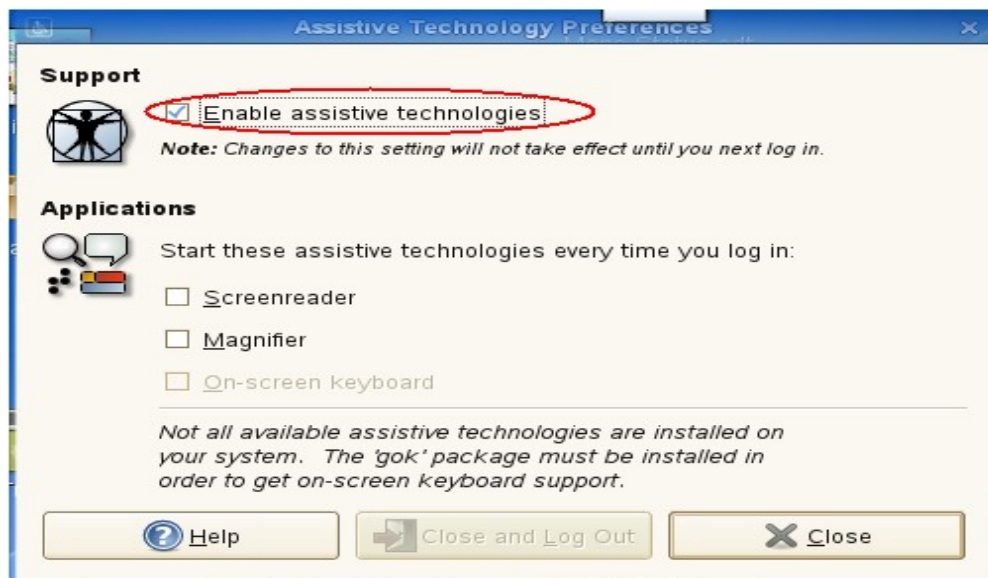
Enabling accessibility

In gnome-control-center, open Assistive Technology.

Drawing 1: Assistive technology preferences under GNOME control center



Drawing 2: Screenshot of Assistive technology preferences dialog



Importing LDTP modules

- `from ldt import *`
`from ldtutils import *`

Reason why we do importing based on the above format instead of 'import ldt' is, if we do the first one, we can just directly use all the ldt functions just by calling their name. If we import the module as 'import ldt', then we need to call the corresponding function as `ldt.<function name>`

Example 1:

`from ldt import *`

Linux Desktop Testing Project - LDTP

```
selectmenuitem ('*-gedit', 'mnuFile;mnuNew')
```

Example 2:

```
import ldtp  
ldtp.selectmenuitem ('*-gedit', 'mnuFile;mnuNew')
```

Call a function to perform an operation

LDTP generally operates on the given object in a particular window.

To select a menu item, you need to call [selectmenuitem](#) function. For example, to select open menu item in gedit application, call the below function

- *selectmenuitem ('frmUnsavedDocument1-gedit', 'mnuFile;mnuOpen')*

When you call the above a new dialog box will be popped up, you can verify whether the window is opened or not either by [guiexist](#) or by [waittillguiexist](#)

- [guiexist](#) function immediately returns either 1 (window exist) or 0 (window does not exist)
[waittillguiexist](#) waits for the window to appear. Wait time out is by default 30 seconds. This default time out can be either increased or decreased using GUI_TIMEOUT

If you want to operate on a push button in a window, you need to call [click](#) function:

To press 'Cancel' button in a GTK Open File Selector dialog

- *click ('dlgOpenFile', 'btnCancel')*

When you do the above operation the GTK File selector dialog disappears. To verify whether the window actually quits or not use [waittillguinotexist](#) function

If you modify any opened file in gedit, the window title will be modified. To continue operating on the window you need to change your context of operation. Reason: As you are aware that LDTP works based on individual window, the context of window will be changed, when the title changes. To overcome this use [setcontext](#) function and when you don't require them use [releasecontext](#)

Edit your current opened file using:

- *settextvalue ('frmUnsavedDocument1-gedit', 'txt0', 'Testing editing')*

This will change the window title. Note, before doing the above operation, title will be 'Unsaved Document 1 - gedit' and after editing the title will look like '*Unsaved Document 1 - gedit'. To further operate on the same window, use

- *setcontext ('Unsaved Document 1 - gedit', '*Unsaved Document 1 - gedit')*

So that you can continue using the same window name, for example:

- *selectmenuitem ('frmUnsavedDocument1-gedit', 'mnuFile;mnuSaveAs')*

The above function will invoke the GTK save dialog box

If any of the action related functions (example: [selectmenuitem](#), [click](#), [settextvalue](#)) fail, an exception is raised. It has to be handled by the program to either continue operating on execution or just halt.

LDTP API

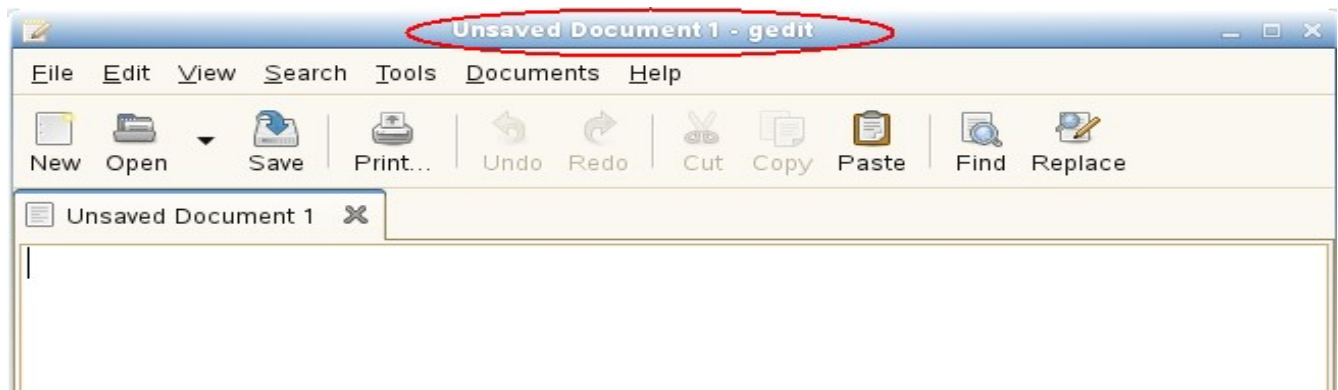
- Refer [LDTP API](#) page for list of implemented LDTP API's

Using / Hacking LDTP

Identifying controls

To operate on a window, first thing we need to know is the window title.

In the following picture you could notice red colored eclipse mark is the window title.



```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtp import *
>>> guixist('*-gedit')
1
>>> guixist('frmUnsavedDocument1-gedit')
1
>>> guixist('frmUnsavedDocument1-*')
1
>>> guixist('frm*-gedit')
1
>>> guixist('Unsaved Document 1 - gedit')
1
>>>
```

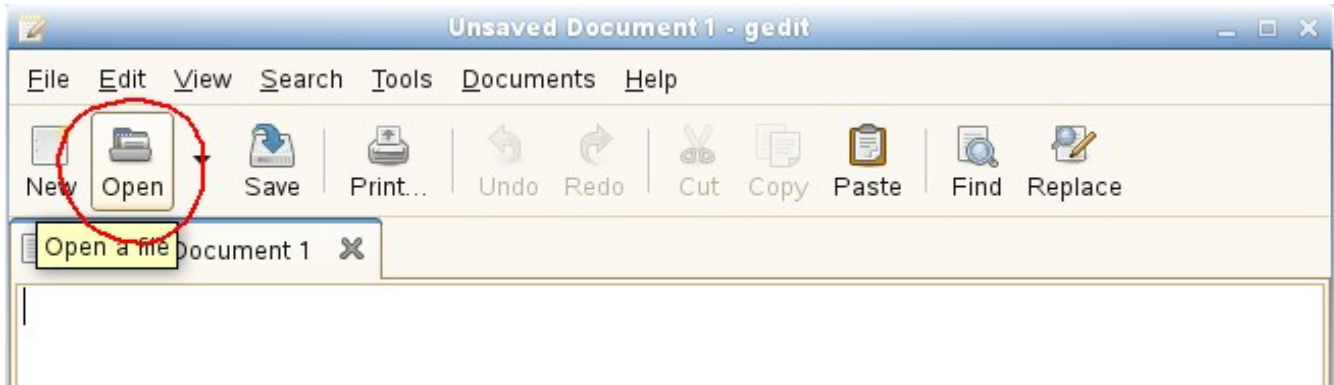
Push button

To operate on an object inside gedit window, we need to know the object information.

To click on open push button in gedit tool bar control, we need to use click API with window name as

Linux Desktop Testing Project - LDTP

first argument and object name as second argument. The above command does the click operation. Informations to be gathered are Window name (Unsaved Document 1 – gedit) and push button control (Open).



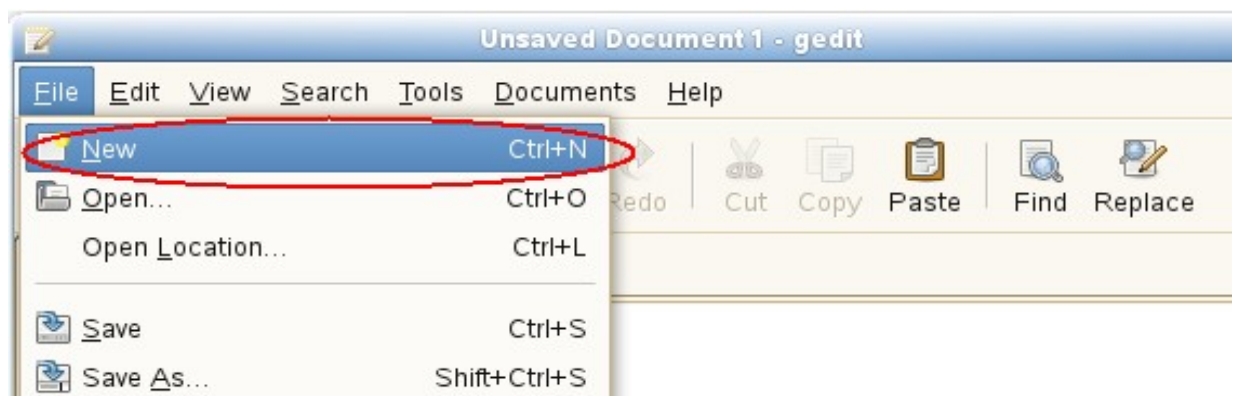
```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtp import *
>>> click ('*-gedit', 'btnOpen')
1
>>>
```

Menu item

To select a menu item under a menu in a window we need to use *selectmenuitem* API.

Informations to be gathered: Window name (Unsaved Document 1 – gedit), menu control (File), menu item control (New).

Incase of menu, we handle them in hierarchy. So, to access 'New' menu item, we need 'File' menu control too.

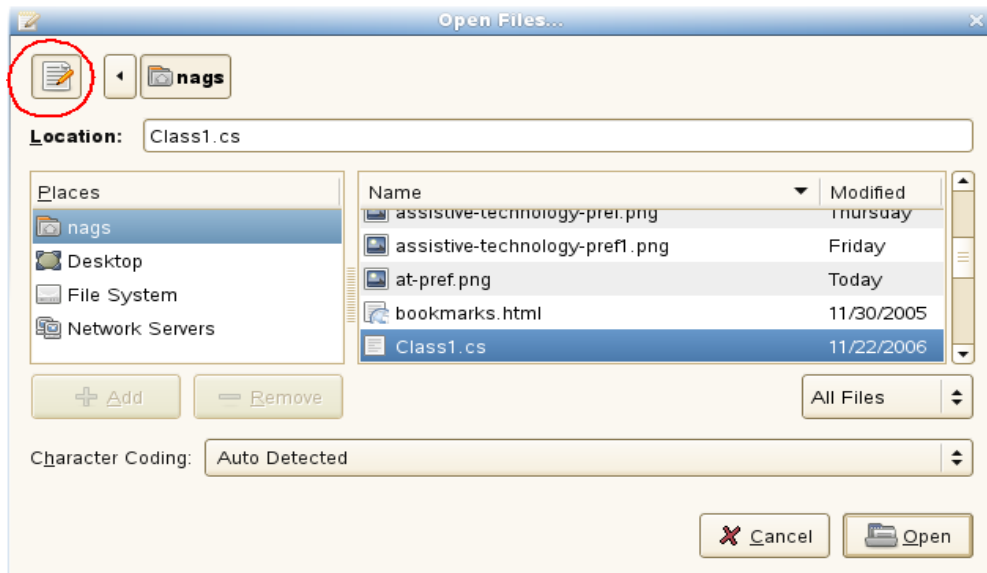


Linux Desktop Testing Project - LDTP

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> selectmenuitem ('*-gedit', 'mnuFile;mnuNew')
1
>>>
```

Toggle button

To operate on a toggle button with a click action, information required are window name (Open Files...) toggle button control (Type a file name).

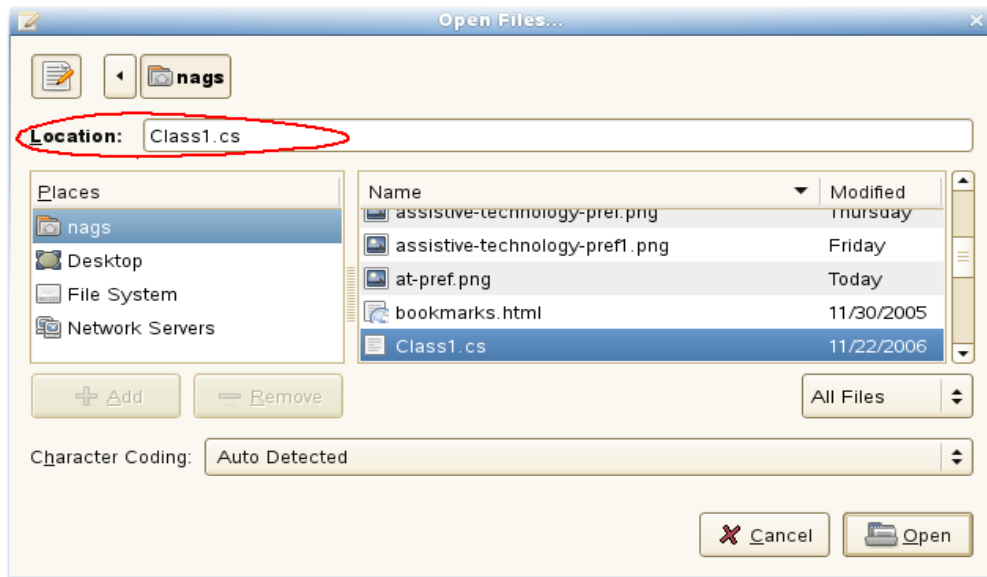


```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> click ('dlgOpenFiles...', 'tbtnTypefilename')
1
>>>
```

Text control

To set a text value in a text box, information like window name (Open Files...), text controls associated label (Location:) and the actual text to be placed (Class1.cs).

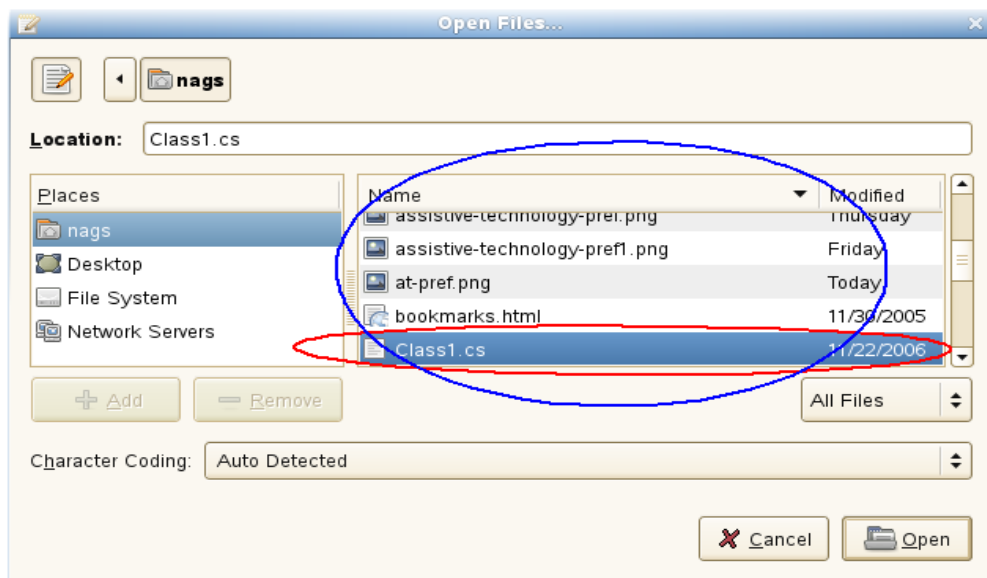
Linux Desktop Testing Project - LDTP



```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> setattr(dlgOpenFiles..., 'txtLocation', 'Class1.cs')
1
>>>
```

Table control

To select a row from the table of GTK open file selector, we need to collect information like, Window name (Open Files...), table name (Files – circled with blue color), row to be selected (Class1.cs).

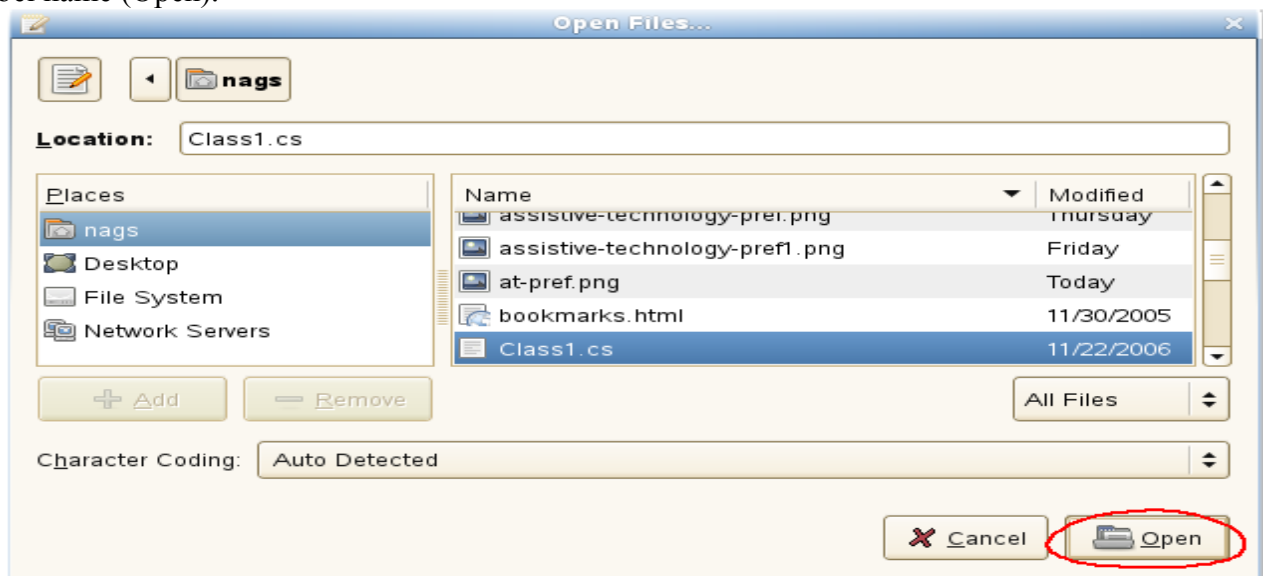


Linux Desktop Testing Project - LDTP

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> selectrow('dlgOpenFiles...', 'tblFiles', 'Class1.cs')
1
>>>
```

Push button

After selecting the file name, to open the file contents, we need to click on Open push button control. For doing this operation we need to gather informations like Window name (Open Files...), push button label name (Open).

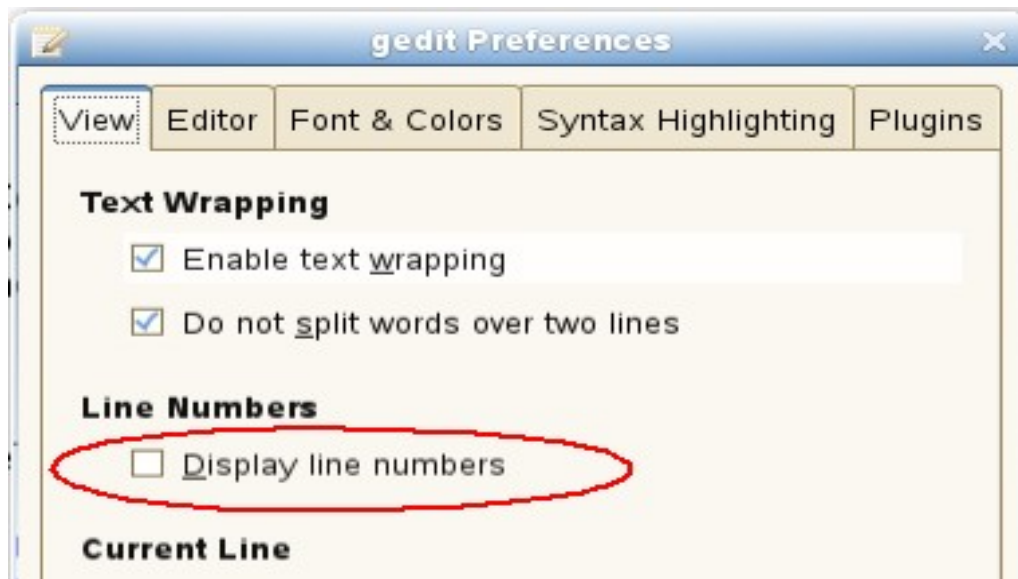


```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> click ('dlgOpenFiles...', 'btnOpen')
1
>>>
```

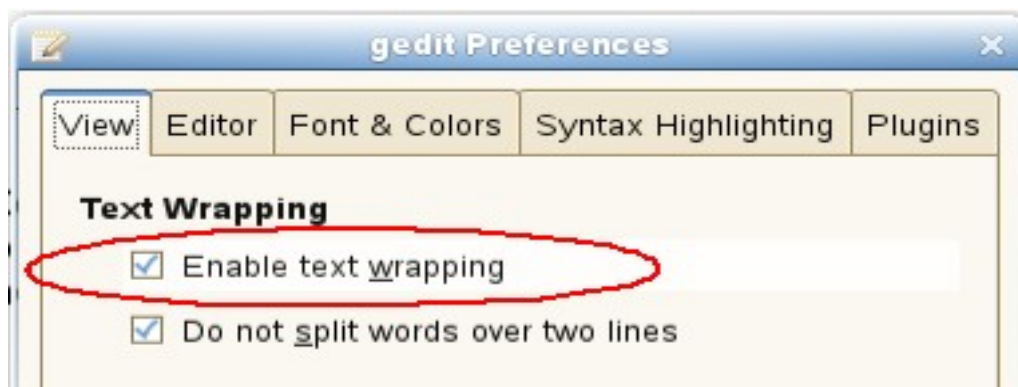
Check box

To click on a check box control, we need to collect informations like window name (gedit Preferences), check box associated label name (Display line numbers).

Linux Desktop Testing Project - LDTP



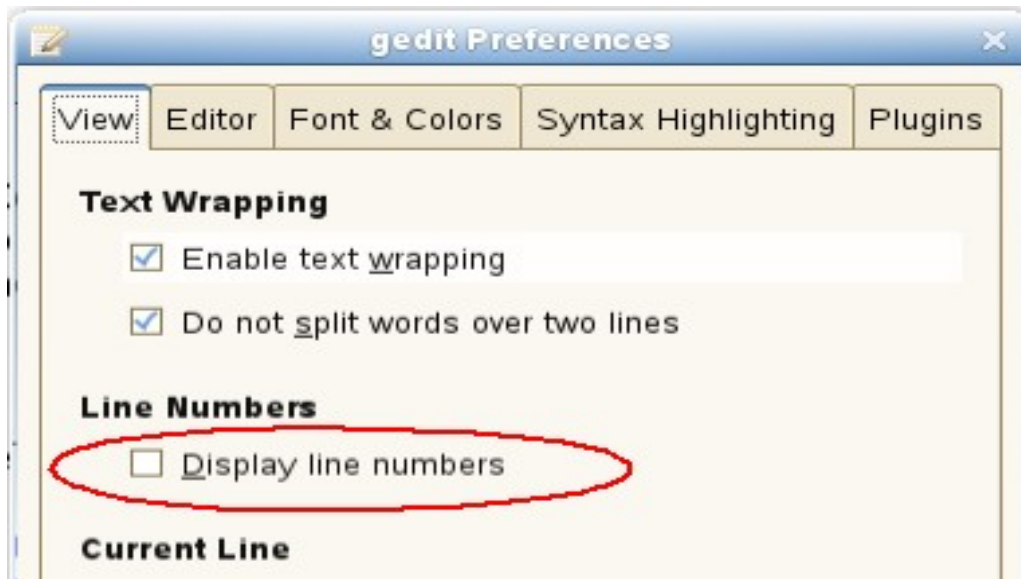
```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> click ('dlggeditPreferences', 'chkDisplaylinenumbers')
1
>>>
```



```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
```

Linux Desktop Testing Project - LDTP

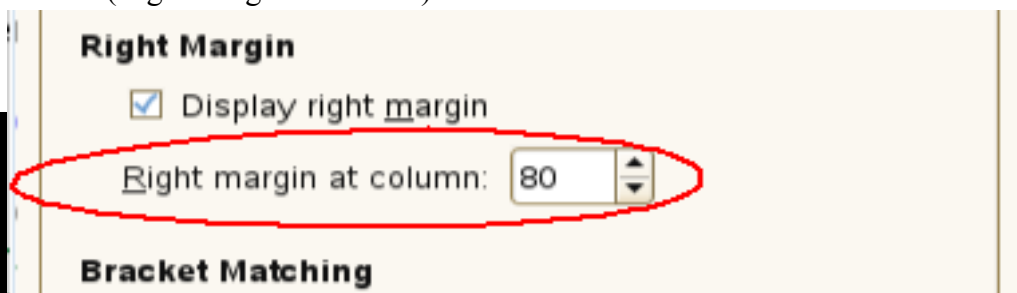
```
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> check ('dlggeditPreferences', 'chkEnabletextwrapping')
1
>>>
```



```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> uncheck ('dlggeditPreferences', 'chkDisplaylinenumbers')
1
>>>
```

Spin button

To operate on a spin button, we need to collect information like Window name (gedit Preferences), spin button control name (Right margin at column).



```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
```

Linux Desktop Testing Project - LDTP

```
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> getvalue('dlggeditPreferences', 'sbtnRightmarginatcolumn')
80.0
>>> setvalue('dlggeditPreferences', 'sbtnRightmarginatcolumn', '81')
1
>>> setvalue('dlggeditPreferences', 'sbtnRightmarginatcolumn', '80')
1
>>>
```

Page tab

To operate on a page tab list, we need to collect information like window name (gedit Preferences), page tab list name (ptl0 in this case, as there are no label or associated label with this page tab list control), page tab name or index starting from 0.

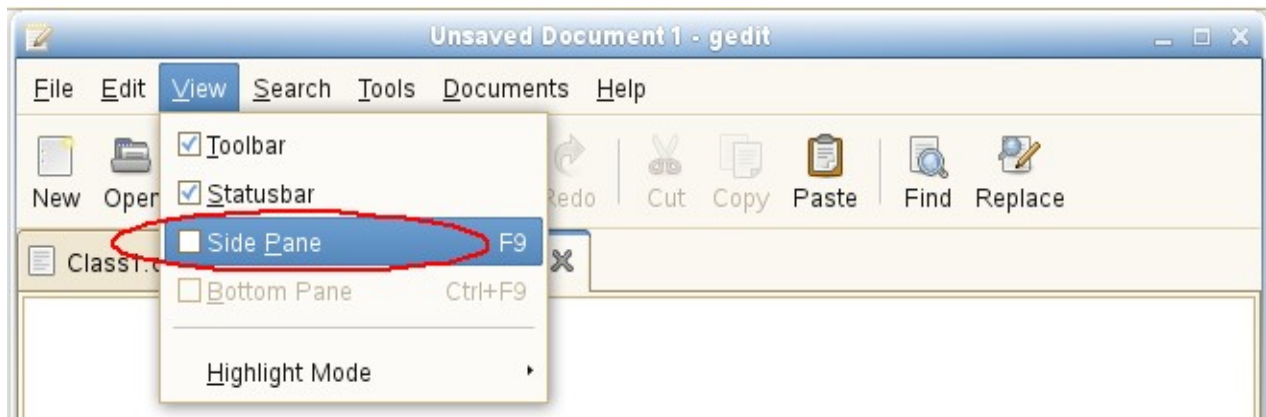


```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> gettabcount ('dlggeditPreferences', 'ptl0')
5L
>>> selecttabindex ('dlggeditPreferences', 'ptl0', 2)
1
>>> selecttab ('dlggeditPreferences', 'ptl0', 'Editor')
1
>>>
```

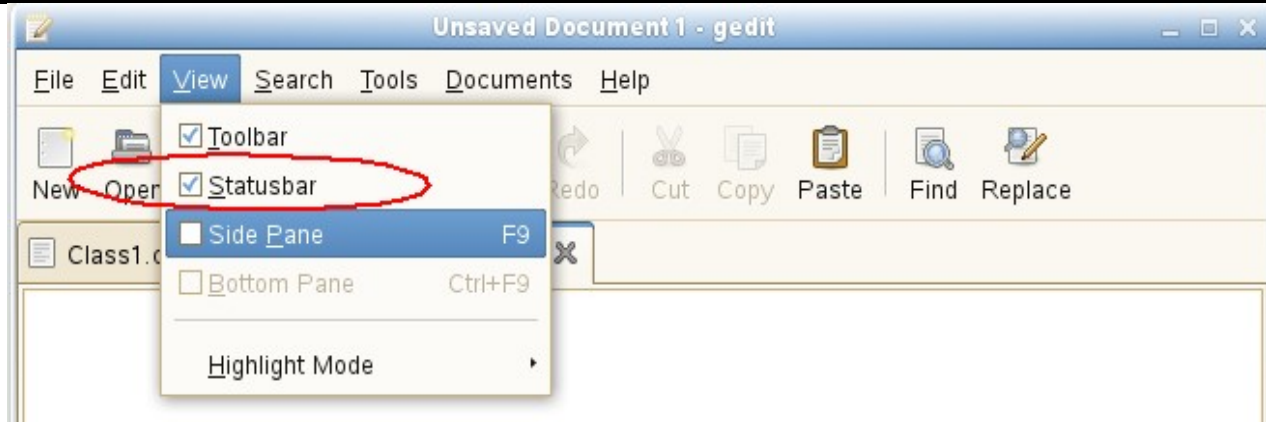
Check menu item

To operate on check menu item, we need to gather information like window name (Unsaved Document 1 – gedit), menu name (View), check menu item name (Side Pane).

Linux Desktop Testing Project - LDTP



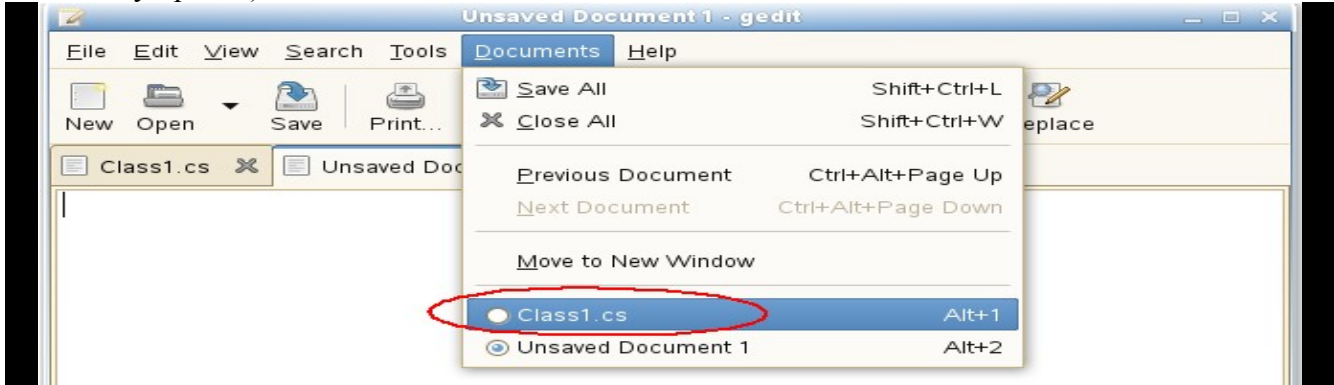
```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> selectmenuItem('*-gedit', 'mnuView;mnuSidePane')
1
>>> menuuncheck('*-gedit', 'mnuView;mnuSidePane')
1
>>>
```



```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> menucheck('*-gedit', 'mnuView;mnuStatusbar')
1
>>>
```

Radio menu item

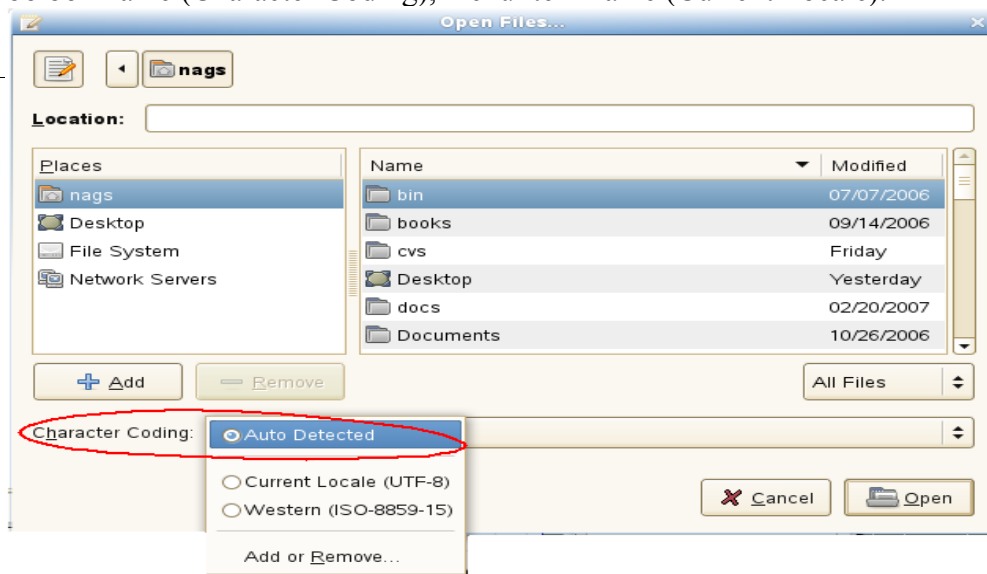
To operate on a radio menu item control, we need to gather informations like window name (Unsaved Document 1 – gedit), menu name (Documents), menu item name (Class1.cs – assuming that Class1.cs is currently opened).



```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtp import *
>>> selectmenuitem ('*-gedit', 'mnuDocuments;mnuClass1.cs')
1
>>> menucheck ('*-gedit', 'mnuDocuments;mnuClass1.cs')
1
>>>
```

Combo box – Menu item

To select a menu item under a combo box, we need to gather informations like window name (Open Files...), combo box name (Character Coding), menu item name (Current Locale).

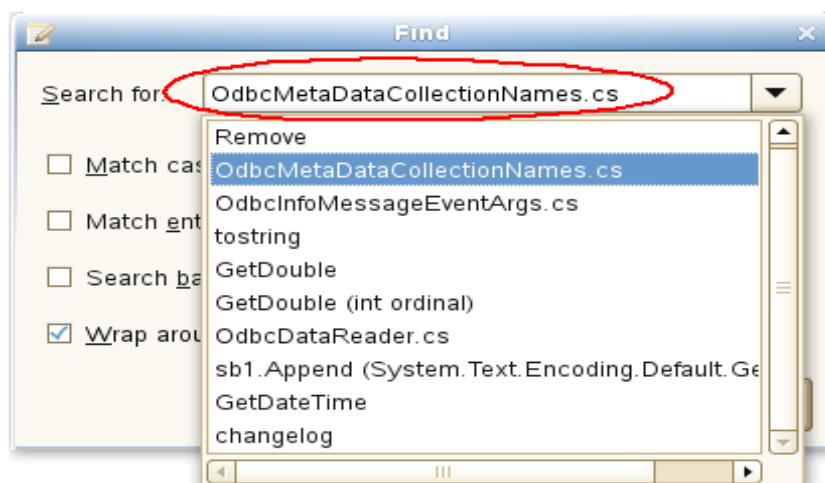
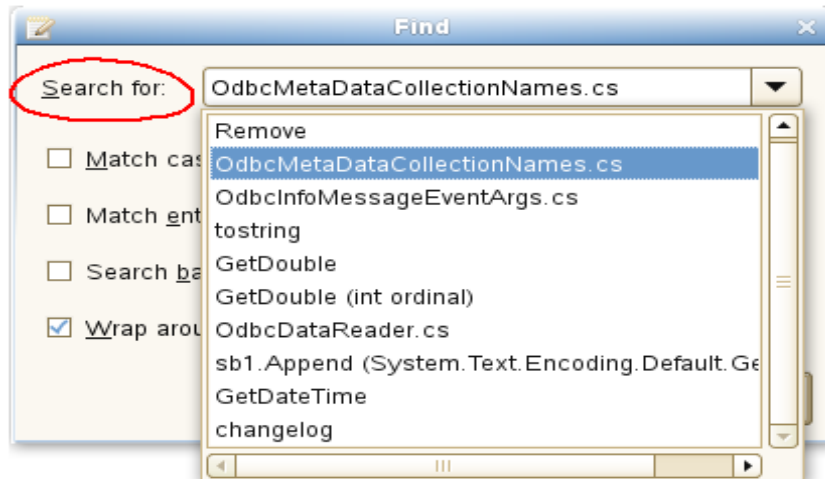


Linux Desktop Testing Project - LDTP

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtp import *
>>> comboselect ('dlgOpenFiles...', 'cboCharacterCoding', 'Current Locale (UTF-8)')
1
>>>
```

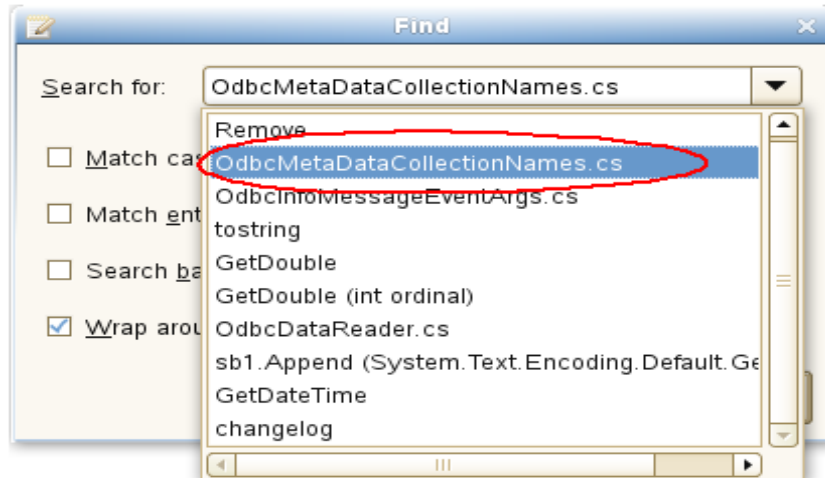
Combo box – List item

To operate on list item under a combo box control, we need to gather informations like window name (Find), Combo box control name (Search for), list item existing content or list item index or new item name (OdbcMetaDataCollectionName.cs)



Linux Desktop Testing Project - LDTP

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtp import *
>>> settextvalue('dlgFind', 'cboSearchfor', 'OdbcMetaDataCollectionNames.cs')
1
>>>
```



```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtp import *
>>> comboboxselect('dlgFind', 'cboSearchfor', 'OdbcMetaDataCollectionNames.cs')
1
>>>
```

Launch application

Application to be tested can be launched using LDTP API `launchapp`.

`launchapp` takes an optional argument, to set the environment variable `GTK_MODULES` and `GNOME_ACCESSIBILITY` and then invoke the respective application. When accessibility is not enabled by default in `gnome-control-center`, this is the preferred way of launching application.

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtp import *
```

```
>>> launchapp ('gedit', 1)
1
>>>
```

Example 1

```
launchapp ('gedit')
```

Example 2

```
launchapp ('gedit', 1)
```

GUI exist

To check a GUI (window) exist, you can use this `guiexist` API. Also it has different flavors like `waittillguiexist`, `waittillguinotexist`.

- `guiexist` function checks whether the given window exists or not. If exist returns 1, else returns 0.
- `waittillguiexist` function will wait for the given window to appear. If appeared returns 1, else returns 0. Difference between `guiexist` and `waittillguiexist` is, `guiexist` returns immediately, but `waittillguiexist` will wait for a max of 30 seconds for a window to appear. **Note:** On doing some operation, if the expected result is, a window will be pop-ed up, then it is recommended to use `waittillguiexist`, instead of `wait` or `sleep`. **Reason:** `wait` or `sleep` will wait till the time period, but `waittillguiexist`, will return immediately once the window appears.
- `waittillguinotexist` function will wait for the given window to close. If closed returns 1, else returns 0. `waittillguinotexist` will wait for a max of 30 seconds for a window to close. **Note:** On doing some operation, if the expected result is, an existing window will be closed, then it is recommended to use `waittillguinotexist`, instead of `wait` or `sleep`. **Reason:** `wait` or `sleep` will wait till the time period, but `waittillguinotexist`, will return immediately once the window closed.

Timeout

GUI timeout

GUI timeout, is the default timeout settings used, by `waittillguiexist` and `waittillguinotexist` functions. This function will wait for the specified number of seconds, for the window to either appear or disappear. Default timeout period is 30 seconds.

This default timeout period that can be modified:

- By setting the environment variable `GUI_TIMEOUT` to whatever seconds.
- By passing a value to `guiTimeout` argument of `waittillguiexist` or `waittillguinotexist` functions.
- By calling `guitimeout` function.
- When invoking LDTP engine, use `-g` option.

Example 1

```
export GUI_TIMEOUT=30
```

Linux Desktop Testing Project - LDTP

Example 2

```
waittillguiexist (*-gedit', guiTimeOut=30)
waittillguinotexist ('dlgOpenFiles...', guiTimeOut=30)
```

Example 3

```
guitimeout (30)
```

Example 4

```
ldtp -g 30
```

OBJ timeout

OBJ timeout, is the default timeout settings used, internally. This function will wait for the specified number of seconds, for the object inside a window to appear. Default timeout period is 5 seconds.

This default timeout period that can be modified:

- By setting the environment variable OBJ_TIMEOUT to whatever seconds.
- By calling objtimeout function.
- When invoking LDTP engine, use -o option.

Example 1

```
export OBJ_TIMEOUT=5
```

Example 2

```
objtimeout (5)
```

Example 3

```
ldtp -o 5
```

Generate raw keyboard events

In some cases, the window we are trying to operate may not be accessibility enabled or we may need to generate non-printable keys (ALT, CTRL, ENTER, BACKSPACE, ESC, F1-F12, SHIFT, CAPS LOCK, TAB, PAGE UP, PAGE DOWN, HOME, END, RIGHT / LEFT / UP / DOWN ARROW KEYS, INS, DEL). We can use generatekeyevent function or enterstring function to simulate the key events, as if the user typed. **Note:** All the non-printable characters will be enclosed with in angular brackets.

Example 1

```
<ctrl>lwww.google.co.in<enter>
```

Example 2

```
<alt><f1>
```

Example 3

```
<control>s
```

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
```

Linux Desktop Testing Project - LDTP

```
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> launchapp ('gedit')
1
>>> waittillguiexist (*-gedit')
1
>>> enterstring ('<alt><tab>')
1
>>> enterstring (*-gedit', 'txt0', '<caps>Testing enterstring API<enter>')
1
>>> generatekeyevent ('<alt><tab>')
1
```

Generate raw mouse events

To generate raw mouse events of different types like, b1c, b1d, b2c, b2d, b3c, b3d, X and Y of screen co-ordinates has to be provided. Here b is button, c is single click, d is double click.

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> generatemouseevent (100, 200) # Default is b1c
1
>>> generatemouseevent (100, 200, 'b1d')
1
```

Application information

On calling `getapplist`, will get all the accessibility application name that are currently running. To get window list for which the application map's are gathered and stored in local cache, use `getwindowlist`. To get all the object list under a window, use `getobjectlist` API. To get a list of properties available under an object, use `getobjectinfo`. To get the property of an object, use `getobjectproperty`.

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> getapplist ()
[u'gnome-session', u'gnome-power-manager', u'gnome-settings-daemon', u'Libbonoboui-Gtk-Module-
init-info', u'nautilus', u'GnomeApplicationBrowser', u'/usr/lib/zen-updater/ZenUpdater.exe', u'gaim',
u'gtk-window-decorator', u'gedit', u'xchat', u'gnome-panel', u'gnome-volume-manager', u'resapplet',
u'nm-applet', u'soffice.bin']
>>> getwindowlist ()
```

```
[u'frmUnsavedDocument1-gedit']
>>> getobjectlist ('*-gedit')
...
>>> getobjectinfo ('*-gedit', 'btnNew')
[u'child_index', u'class', u'description', u'parent', u'label']
>>> getobjectproperty ('*-gedit', 'btnNew', 'class')
'New'
```

Callback – On new window creation

Register a callback event, when a window with given title is created. Glob type pattern can be given as title name.

Advantage

Unexpected window can be easily handled using this. For example, the password dialog box of Evolution, connection reset by peer dialog, application crash dialog, etc.

Example

```
from ldtp import *
import threading
callbackRunning = threading.Event ()
callbackRunning.clear ()
callbackState = threading.Event ()
callbackState.clear ()
def cb ():
    callbackState.set ()
    waittillguiexist ('dlgReplace')
    click ('dlgReplace', 'btnClose')
    callbackState.clear ()
    callbackRunning.set ()
    print 'callbackend'
onwindowcreate ('Replace', cb)
click ('*gedit', 'btnReplace')
click ('*gedit', 'btnOpen')
waittillguiexist ('dlgOpenFiles...')
click ('dlgOpenFiles...!', 'btnClose')
if callbackState.isSet ():
    print 'Waiting for callback to complete'
    callbackRunning.wait ()
    print 'callbackset'
print 'test end'
```

Color coding

Light yellow – Thread creation

Light green – Callback definition

Red – Callback registration

Magenta – General operation, which will invoke a window

Light blue – Wait for callback to complete, if invoked.

Logging

```
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtplib import *
>>> log ('test script', 'debug')
1
>>> log ('test script', 'warning')
1
>>> log ('test script', 'error')
1
>>> log ('test script', 'cause')
1
>>>
```

Example script

```
from ldtplib import *
from ldtputils import *

try:
    launchapp ('gedit')
    if waittillguiexist ('*-gedit') == 0:
        raise LdtplibExecutionError ('Gedit window does not exist')
    selectmenuitem ('*-gedit', 'mnuFile;mnuOpen')
    if waittillguiexist ('dlgOpenFiles') == 0:
        raise LdtplibExecutionError ('Open Files dialog does not exist')
    selectrow ('dlgOpenFiles...', 'tblFiles', fileName [0])
    click ('dlgOpenFiles...', 'btnOpen')
    if waittillguinotexist ('dlgOpenFiles') == 0:
        raise LdtplibExecutionError ('Open Files dialog still exist')
except LdtplibExecutionError, msg:
    raise
```

How to execute LDTP scripts

Make sure that you have the following files in current working directory

- You need to have test scripts to be executed

Invoking python script

```
$ python <script file name.py>
```

Example

Linux Desktop Testing Project - LDTP

```
python gedit.py
```

Suggestions from LDTP team

- When a new window is expected after an operation, we suggest to use `waittillguiexist` and on some operation, if a window is expected to close we suggest to use `waittillguinotexist`. In both cases, the time-out period is 30 seconds. This value can be modified – refer LDTP API reference.

How to operate LDTP from a remote system

LDTP engine

Follow one of the options to start LDTP engine (ldtp binary) in the remote box

Option 1

```
$ldtp -s
```

Option 2

```
$ ldtp -p <port number to start> # Default port number is 4118
```

LDTP client

Follow one of the options in the client side to communicate to LDTP engine

Option 1

```
export LDTP_SERVER_ADDR=host-name or ip address
```

```
export LDTP_SERVER_PORT=<port number to communicate, as mentioned in LDTP engine>
```

```
python <script file name>.py or ldtprunner test-runner.xml
```

Option 2

```
export LDTP_SERVER_ADDR=host-name or ip address
```

```
python <script file name>.py or ldtprunner test-runner.xml # This will use default port number.
```

Troubleshooting LDTP

In-case, if you want to see whats happening on executing some LDTP commands, follow the steps

In a terminal

```
$ export LDTP_DEBUG=2 # If bash shell
$ ldtp
Client packet len: 82
i = 0
Data read 82, packet-len = 82, bytes read = 82, data: <?xml version="1.0"?
<<REQUEST><ACTION>124</ACTION><ID>MainThread124</ID></REQUEST>
PACKET LENGTH: 0
Received packet [<?xml version="1.0"?
<<REQUEST><ACTION>124</ACTION><ID>MainThread124</ID></REQUEST>] through 15
Node: ACTION
action_name: 124
Node: ID
request id: MainThread124
```

Linux Desktop Testing Project - LDTP

```
Command: 124
Accessible application name: Thunderbird
Accessible application name: gnome-panel
Accessible application name: xchat
Accessible application name: nm-applet
Accessible application name: nautilus
Accessible application name: gaim
Accessible application name: acroread
Accessible application name: soffice.bin
Accessible application name: gtk-window-decorator
Accessible application name: gedit
LIST: <?xml version="1.0" encoding="utf-8"?
<<OBJECTLIST><OBJECT>nautilus</OBJECT><OBJECT>gaim</OBJECT><OBJECT>gtk-
window-
decorator</OBJECT><OBJECT>gedit</OBJECT><OBJECT>xchat</OBJECT><OBJECT>gnome-
panel</OBJECT><OBJECT>Thunderbird</OBJECT><OBJECT>nm-
applet</OBJECT><OBJECT>soffice.bin</OBJECT><OBJECT>acroread</OBJECT></OBJECTLIS
T>
resp_len = 117
Sending..
538
Response packet: <?xml version="1.0" encoding="utf-8"?
<<RESPONSE><ID>MainThread124</ID><STATUS><CODE>0</CODE><MESSAGE>Successfull
y completed</MESSAGE></STATUS><DATA><LENGTH>325</LENGTH><VALUE><!
[CDATA[<?xml version="1.0" encoding="utf-8"?
<<OBJECTLIST><OBJECT>nautilus</OBJECT><OBJECT>gaim</OBJECT><OBJECT>gtk-
window-
decorator</OBJECT><OBJECT>gedit</OBJECT><OBJECT>xchat</OBJECT><OBJECT>gnome-
panel</OBJECT><OBJECT>Thunderbird</OBJECT><OBJECT>nm-
applet</OBJECT><OBJECT>soffice.bin</OBJECT><OBJECT>acroread</OBJECT></OBJECTLIS
T>]]</VALUE></DATA></RESPONSE>
Msg:
Bytes sent: 542
```

In another terminal

```
$ export LDTP_DEBUG=2 # If bash
nags@nags:~> python
Python 2.5 (r25:51908, Nov 25 2006, 15:39:45)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ldtp import *
>>> getapplist()
124 ()
Send packet <?xml version="1.0"?
<<REQUEST><ACTION>124</ACTION><ID>MainThread124</ID></REQUEST>
```

Linux Desktop Testing Project - LDTP

```
Received packet size 538
Received response Packet <?xml version="1.0" encoding="utf-8"?
><RESPONSE><ID>MainThread124</ID><STATUS><CODE>0</CODE><MESSAGE>Successful
y completed</MESSAGE></STATUS><DATA><LENGTH>325</LENGTH><VALUE><![
CDATA[<?xml version="1.0" encoding="utf-8"?
><OBJECTLIST><OBJECT>nautilus</OBJECT><OBJECT>gaim</OBJECT><OBJECT>gtk-
window-
decorator</OBJECT><OBJECT>gedit</OBJECT><OBJECT>xchat</OBJECT><OBJECT>gnome-
panel</OBJECT><OBJECT>Thunderbird</OBJECT><OBJECT>nm-
applet</OBJECT><OBJECT>soffice.bin</OBJECT><OBJECT>acroread</OBJECT></OBJECTLIS
T>]]></VALUE></DATA></RESPONSE>
[u'nautilus', u'gaim', u'gtk-window-decorator', u'gedit', u'xchat', u'gnome-panel', u'Thunderbird', u'nm-
applet', u'soffice.bin', u'acroread']
>>>
```

Bibliography

http://en.wikipedia.org/wiki/Software_testing

<http://safsdev.sf.net>