

# Diagram tcl package guide

---

A pure tcl diagram package implemented on the tk canvas

José E. Marchesi ([jemarch@gnu.org](mailto:jemarch@gnu.org))

---

Diagram tcl package guide, version 1.0.0

Copyright © 2005 José E. Marchesi.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being “GNU General Public License”, the Front-Cover texts being (a) (see below), and with the Back-Cover Texts being (b) (see below). A copy of the license is included in the section entitled “GNU Free Documentation License”.

a. “Diagram tcl package guide”

b. “Your have freedom to copy and modify this Manual, like GNU software.”

## Table of Contents

List of Figures .....	1
1 Diagram Overview .....	2
2 Diagram manipulation .....	3
3 Object manipulation .....	4
4 Port manipulation .....	6
5 Connector manipulation.....	7

## List of Figures

Figure 1.1: Diagram Package Architecture .....	2
--	---

# 1 Diagram Overview

The Diagram package is a pure Tcl implementation that uses the Tk canvas in order to implement interactive diagrams.

The Tk canvas allow to construct many kind of structured graphics. That means that almost every kind of user interface can be constructed using the Tk canvas. GNU Ferret is mainly a graphical editor, with well defined edited objects: entities, relationships, etc. Each of these objects usually need a graphical representation with which the user interact.

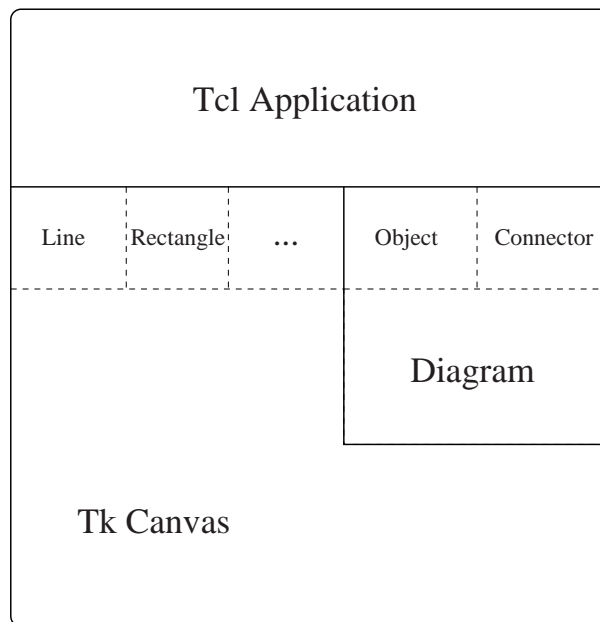


Figure 1.1: Diagram Package Architecture

A diagram is composed by two main component types:

- Objects with shape.
- Connectors that involves one or more objects.

## 2 Diagram manipulation

<code>diagram::create</code> <i>DNAME WIDGET</i>	[Function]
Creates a new diagram named <i>DNAME</i> on <i>WIDGET</i> .	
<code>diagram::destroy</code> <i>DNAME</i>	[Function]
Destroy <i>DNAME</i> .	
<code>diagram::pack</code> <i>DNAME</i>	[Function]
Pack the tk widgets of <i>DNAME</i> .	
<code>diagram::unpack</code> <i>DNAME</i>	[Function]
Unpack the tk widgets of <i>DNAME</i> .	
<code>diagram::get_object_list</code> <i>DNAME</i>	[Function]
Return a list with the names of all the objects present on <i>DNAME</i> .	
<code>diagram::get_connector_list</code> <i>DNAME</i>	[Function]
Return a list with the names of all the connectors present on <i>DNAME</i> .	
<code>diagram::print_ps</code> <i>DNAME FILENAME PAPERTYPE COLORMODE</i> <i>PAGEANCHOR</i>	[Function]
Where,	
<i>colormode</i> color or gray	
<i>papertype</i> fit or a4	

### 3 Object manipulation

<code>diagram::exist_object</code> <i>DNAME ONAME</i>	[Function]
Return 1 if <i>ONAME</i> is an existing object into <i>DNAME</i> .	
<code>diagram::set_object</code> <i>DNAME ONAME NEWOBJ</i>	[Function]
Set <i>NEWOBJ</i> as the new object structure for <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::get_object</code> <i>DNAME ONAME</i>	[Function]
Return the object structure for <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::create_object</code> <i>DNAME ONAME OTYPE LOCATION</i>	[Function]
Creates a new object on the diagram named <i>ONAME</i> , of type <i>OTYPE</i> at the point <i>LOCATION</i> .	
<code>diagram::remove_object</code> <i>DNAME ONAME</i>	[Function]
Remove <i>ONAME</i> from the <i>DNAME</i> diagram.	
<code>diagram::set_object_location</code> <i>DNAME ONAME LOCATION</i>	[Function]
Set <i>LOCATION</i> as the new location point of <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::get_object_location</code> <i>DNAME ONAME</i>	[Function]
Return the location of <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::set_object_window</code> <i>DNAME ONAME WINDOW</i>	[Function]
Set <i>WINDOW</i> as the new window for <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::get_object_window</code> <i>DNAME ONAME</i>	[Function]
Get the window from <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::get_object_type</code> <i>DNAME ONAME</i>	[Function]
Return the type of <i>ONAME</i> in <i>DNAME</i> .	
<code>diagram::set_object_attribute</code> <i>DNAME ONAME ANAME VALUE</i>	[Function]
Set <i>VALUE</i> as the new value for <i>ANAME</i> on <i>ONAME</i> in <i>DNAME</i> .	
<code>diagram::get_object_attribute</code> <i>DNAME ONAME ANAME</i>	[Function]
Return the value of <i>ANAME</i> from <i>ONAME</i> in <i>DNAME</i> .	
<code>diagram::set_object_attributes</code> <i>DNAME ONAME ATTRIBUTES</i>	[Function]
Set <i>ATTRIBUTES</i> as the new attribute set for <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::get_object_attributes</code> <i>DNAME ONAME</i>	[Function]
Return the attributes structure from <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::get_object_connectors</code> <i>DNAME ONAME</i>	[Function]
Return a list with the name of all the connectors that leads to <i>ONAME</i> on <i>DNAME</i> .	
<code>diagram::set_object_connectors</code> <i>DNAME ONAME CNNS</i>	[Function]
Set <i>CNNS</i> as the new connector list for <i>OBJECT</i> .	

<code>diagram::object_connector_exist</code> <i>DNAME ONAME CNN</i>	[Function]
Return 1 if CNN is on the ONAMEs connector list. Return 0 else.	
<code>diagram::add_connector_to_object</code> <i>DNAME ONAME CNN</i>	[Function]
Add CNN to the ONAMEs connector list.	
<code>diagram::remove_connector_from_object</code> <i>DNAME ONAME CNN</i>	[Function]
Remove CNN from the ONAMEs connector list.	
<code>diagram::update_object</code> <i>DNAME ONAME</i>	[Function]
Update the geometry and the connectors of ONAME.	

## 4 Port manipulation

<code>diagram::create_port</code> <i>POINT ORIENT</i>	[Function]
Create a new port.	
<code>diagram::get_port_point</code> <i>PORT</i>	[Function]
<code>diagram::set_port_point</code> <i>PORT POINT</i>	[Function]
<code>diagram::get_port_orient</code> <i>PORT</i>	[Function]
<code>diagram::set_port_orient</code> <i>PORT ORIENT</i>	[Function]

## 5 Connector manipulation

<code>diagram::get_connector</code> <i>DNAME CNAME</i>	[Function]
Return the structure for CNAME on DNAME.	
<code>diagram::set_connector</code> <i>DNAME CNAME CNN</i>	[Function]
Set CNN as the new structure for CNAME on DNAME.	
<code>diagram::exist_connector</code> <i>DNAME CNAME</i>	[Function]
Return 1 if CNAME is in DNAME. Return 0 else.	
<code>diagram::get_connector_obj1</code> <i>DNAME CNAME</i>	[Function]
<code>diagram::set_connector_obj1</code> <i>DNAME CNAME ONAME</i>	[Function]
<code>diagram::get_connector_port1</code> <i>DNAME CNAME</i>	[Function]
<code>diagram::set_connector_port1</code> <i>DNAME CNAME PORT</i>	[Function]
<code>diagram::get_connector_obj2</code> <i>DNAME CNAME</i>	[Function]
<code>diagram::set_connector_obj2</code> <i>DNAME CNAME ONAME</i>	[Function]
<code>diagram::get_connector_port2</code> <i>DNAME CNAME</i>	[Function]
<code>diagram::set_connector_port2</code> <i>DNAME CNAME PORT</i>	[Function]
<code>diagram::get_connector_drawproc</code> <i>DNAME CNAME</i>	[Function]
<code>diagram::set_connector_drawproc</code> <i>DNAME CNAME DRAWPROC</i>	[Function]
<code>diagram::get_connector_cp</code> <i>DNAME CNAME</i>	[Function]
<code>diagram::set_connector_cp</code> <i>DNAME CNAME CP</i>	[Function]
<code>diagram::create_connector</code> <i>DNAME CNAME OBJ1 OBJ2</i> <i>DRAW_PROC</i>	[Function]
Create a new connector named CNAME that connect OBJ1 and OBJ2, with DRAW_PROC as the custom drawing procedure.	
<code>diagram::update_connector</code> <i>DNAME CNAME</i>	[Function]
Update the geometry and paint CNAME.	
<code>diagram::remove_connector</code> <i>DNAME CNAME</i>	[Function]
Remove the connector named CNAME from DNAME.	