Example

```
- include 'tabdialog.g'
- tt := tabdialog()
```

1.4.39 tapedeck - Tool

Package	**display**
Module	**widgets**

Glish/Tk widget providing tapedeck-like control buttons
include "tapedeck.g"

Constructors

tapedeck	*Construct a tapedeck*

Functions

disable	*Disable the tapedeck controls*
dismiss	*Disable and dismiss the tapedeck*
done	*Disable and dismiss the tapedeck*
enable	*Enable the tapedeck controls*

Description

This Glish/Tk widget is simply a collection of upto seven buttons which can be used wherever operations like "play", "rewind" and "stop" are required. The constructor offers flexibility in which buttons are made available, and what colors they will be. An event is emitted whenever a button is pressed, and is one of: `tostart`, `reversestep`, `reverseplay`, `stop`, `forwardplay`, `forwardstep`, `toend`.

tapedeck - Constructor

Package	**display**
Module	**widgets**
Tool	**tapedeck**

Construct a tapedeck

Synopsis

tapedeck(parent, background, hasstop, hasplay, hasstep, hasto, hasforward, hasreverse, stopcolor, playcolor, stepcolor, tocolor, orient, widgetset)

Arguments

parent	*Parent of the menu* *Allowed: Tk frame*
background	*Background color* *Allowed: X color* *Default: 'lightgrey'*
hasstop	*Whether to provide 'stop' button* *Allowed: Boolean* *Default: T*
hasplay	*Whether to provide 'play' button/s* *Allowed: Boolean* *Default: T*
hasstep	*Whether to provide 'step' button/s* *Allowed: Boolean* *Default: T*
hasto	*Whether to provide 'to' button/s (ie. to end, to start)* *Allowed: Boolean* *Default: T*
hasforward	*Whether to provide forward-direction buttons* *Allowed: Boolean* *Default: T*
hasreverse	*Whether to provide reverse-direction buttons* *Allowed: Boolean* *Default: T*
stopcolor	*Color of the 'stop' button* *Allowed: X color* *Default: 'black'*
playcolor	*Color of the 'play' button/s* *Allowed: X color* *Default: 'black'*
stepcolor	*Color of the 'step' button/s* *Allowed: X color* *Default: 'black'*
tocolor	*Color of the 'to' (cue) button/s* *Allowed: X color* *Default: 'black'*
orient	*Orientation of the tapedeck* *Allowed: 'horizontal' 'vertical'* *Default: 'horizontal'*
widgetset	*Widget set to use* *Allowed: A widgetserver closure tool* *Default: dws*

Returns
tapedeck widget

Description
Constructs a tapedeck according to the specified construction arguments, and returns a ref to the widget.

Example

```
include 'tapedeck.g'
f := frame();
td := tapedeck(f, background='black', hasto=F);
```

---

tapedeck.dismiss - Function

Package	**display**
Module	**widgets**
Tool	**tapedeck**

Disable and dismiss the tapedeck

Synopsis
dismiss()

Returns
None

Description
This provides a tidy way to remove a tapedeck once it is finished with. This is now deprecated - use 'done' instead.

Example

```
f := frame();
td := tapedeck(f);
# ... fill in whenevers ...
await td->stop;
td.dismiss();
```

---

tapedeck.done - Function

Package	**display**
Module	**widgets**
Tool	**tapedeck**

Disable and dismiss the tapedeck

Synopsis
done()

Returns
None

Description
This provides a tidy way to remove a tapedeck once it is finished with.

Example

```
f := frame();
td := tapedeck(f);
# ... fill in whenevers ...
await td->stop;
td.done();
```

---

tapedeck.enable - Function

Package	**display**
Module	**widgets**
Tool	**tapedeck**

Enable the tapedeck controls

Synopsis
enable()

Returns

None

Description
This will enable the tapedeck buttons, provided it is balanced by an equal number of calls to disable.

Example

```
f := frame();
td := tapedeck(f);
td.disable();
# ... fill in some whenevers
td.enable();
```

---

tapedeck.disable - Function

Package	**display**
Module	**widgets**
Tool	**tapedeck**

Disable the tapedeck controls

Synopsis
disable()

Returns
None

Description
This will disable the tapedeck buttons, provided it is balanced by an equal number of calls to enable.

Example

```
f := frame();
td := tapedeck(f);
```