```
td.disable();
# ... fill in some whenevers
td.enable();
```

1.4.40 taqlwidget - Tool

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

Widget to form a TaQL command
include "taqlwidget.g"

Description

The Table Query Language (TaQL) is an SQL-like language for defining commands to do selections and/or sorts on an AIPS++ table. It offers a lot of power and flexibility, but the syntax is not always clear. Furthermore, the names and data types of the columns in the table might not be known. This makes it sometimes hard to create a query command.
The TaQL widget is a Query-By-Example like widget to assist the user in creating a TaQL command. Such a command consists of four optional parts. Constructor arguments control if the user can specify these parts by enabling the appropriate parts of the widget.

- *The select-list tells which columns to select from the table.*
  A check box for each column name lets the user define the select-list. Pressing a button means that the column is added to the select-list in the entry box at the bottom. Pressing again removes it. It is also possible to edit the select-list directly.
  Note that an empty select-list means that all columns are selected.
- *The where-expression tells which rows to select from the table.*
  This part of the widget is always active and is meant to ease the process of creating the where-expression. The available columns and, if applicable, lists of possible values are shown in a form. The popuphelp shows the data type, etc.. The user can fill in the form to create (part of) the where-expression. Special buttons are available to create more complex where-expressions with parentheses and AND, OR, and NOT operators by combining the current contents of the form with the where-expression already present in the entry box at the bottom. It is also possible to edit the where-expression in the entry box directly.
  The expand button is only giving information. Pressing it shows how the current contents of the form convert to a where-expression

- *The sort-list tells if and how the resulting table is sorted.*
  A listbox is shown for each column to let the user add or remove the column to the sort-list in the entry box at the bottom. The user can choose No Sort, Ascending, or Descending. The No Sort choice removes the column from the sort-list. It is also possible to edit the sort-list directly.
- *The giving entry box lets the user specify the name of the resulting table. No name means that the result is transient.*

All these parts can be filled in manually, but it is easier to use the form at the top. It shows all available fields and if appropriate also a check box to tell if the field has to added to select list. It can also have a sort menu to add the field to the sort list.
Each field is followed by an operator menu and an entry box which can be used to fill in values. Multiple values are possible separated by commas. When a field can only have a limited set of values, the entry box is preceeded by a menu from which the user can select values. Fields of type string have some extra operators to compare the field with a regular expression (or filename-like pattern). It also has an option menu to make tell if case has to be ignored (default no) and if the string has to be enclosed in quotes (default yes). Note that in a TaQL command an unquoted string is treated as the name of another field.
To make the widget appear less overwhelming, the more advanced features (like the OR button and the resulting entry boxes) are hidden in a rollup. The rollup also has an expand button to show what the TaQL string looks like for the current contents of the form.
Constructing the widget is driven by the description of the table columns as given by the `table` *function getdesc().*
However, use of the widget is not limited to tables. As long as a correct description is given, the widget can be used to create query commands for other purposes as well.

taqlwidget - Constructor

Package	**display**
Module	**widgets**
Tool	**taqlwidget**

Construct TaQL widget

Synopsis
taqlwidget(cdesc, title, canselect, cansort, cangiving, heigth, giving, widgetset)

Description
This constructor creates the widget from the given description. The widget runs

asynchronously as a subsequence. When the user presses the `Go` or `Cancel` button a `returns` event is generated which gives a record containing the fields `select`, `where`, `orderby`, and `giving`. These fields represent the four parts of a full TaQL command. All fields are empty if the `Cancel` button was pressed.
The constructor fails if no GUI is active in the system.
The argument `cdesc` determines the columns to be shown. It is a record in which each field describes a column. The field name is the column name; the field value is a record from which the following fields are used (if present):

- *`valueType` defines the type. Possible values are string, integer, float, double, complex, dcomplex, and boolean. Its main use is in building the help info.*
  If the type is string a value is enclosed in quotes if necessary.
- *`istime` defines if the field contains a date/time (defaults to F). If so, some extra operations will be added automatically to handle the date/time correctly. If the valueType is string, the DATETIME function is applied to the field to convert the string to a date/time. Otherwise the field is a numeric date/time in seconds and is divided by 24*3600 to convert it to days. The MJD function is applied to the value to convert it to a double value.*
- *`ndim` defines the dimensionality of an array where 0 or -1 indicates any dimensionality. If the field is not present, the column is a scalar. It is only used in building the help info.*
- *`shape` defines the fixed shape of arrays in a column. It is only used in building the help info.*
- *`comment` can be used to give some extra information about the column. It is only used in building the help info.*
- *`labels` defines the possible values for the column if it is known that the column can only have a limited number of values. Those values are shown in an optionmenu where the user can select from. It should be a vector of type `valueType`.*
- *`labelnames` gives symbolic names for the possible values defined in `labels`. They are also shown in the optionmenu. They are only used if `labels` is given and if both vectors have the same length.*

The table description returned by table.getdesc() is a record with the fields shown above. Only the fields `istime`, `labels`, and `labelnames` are not part of it. It means that a table description can directly be used in the `taqlwidget` constructor.
The description can contain as many fields as one likes. When more fields than the widget heigth (argument `height`) are given, a vertical scrollbar is added which can be used to scroll through all fields.

The arguments `canSomething` determine if the user is allowed to build a select string, sort string, and giving string. If so, the widget is extended with the appropriate entry boxes, etc. to make it possible.

Arguments

cdesc	*in*	*The column descriptions* *Allowed: Record*
title	*in*	*Title string* *Allowed: String* *Default: Query Dialogue*
canselect	*in*	*Possible to build select string?* *Allowed: Bool* *Default: F*
cansort	*in*	*Possible to build sort string?* *Allowed: Bool* *Default: F*
cangiving	*in*	*Possible to build giving string?* *Allowed: Bool* *Default: F*
heigth	*in*	*Height of the field list* *Allowed: integer* *Default: 12*
giving	*in*	*default giving string* *Allowed: string* *Default: ''*
widgetset	*in*	*widgetset to be used* *Allowed: widgetset tool* *Default: dws*

Returns

subsequence

Example

```
include 'table.g'
include 'taqlwidget.g'
t := table('3C273XC1.MS');
q := taqlwidget(t.getdesc(), cansort=T);
await q->returns;
rec := $value;
t1 := t.query (rec.where, rec.giving, rec.orderby, rec.select);
```