fiasco

Nils Bandener

COLLABORATORS			
	TITLE : fiasco		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Nils Bandener	February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

fiasco

1.1 Fiasco.guide

```
                Fiasco  Release 1.2
Copyright © 1995-1996 Nils Bandener
```

1.2 Legal Things

Legal Things

The Program "Fiasco" and associated files, hereafter called Fiasco, are provided "as is". No representations or warranties are made regarding to accuracy, reliability or correctness of Fiasco, either expressed or implied. In no case am I responsible for any damages caused by this software.

Fiasco is not Public Domain. I reserve all rights.
Fiasco Copyright © 1995-1996 Nils Bandener.

Fiasco may be redistributed under the following conditions:

- The program package has to be complete. See the file list for a complete listing of all files that comprise Fiasco 1.2.

- Fiasco may not be distributed for commercial purposes without a written permission by the author. This includes the distribution of Fiasco for excessively high prices. You may only charge a small fee for media and copying. The distribution on CD-Roms is allowed, if the price of the CD-Rom is not higher than the price of the "Fresh Fish" CD-Roms of Fred Fish. Distribution on cover disks or cover CDs of magazines is allowed, if the price of the magazine is less than USD 10 or DM 12 in the case of floppy disks or USD 12 or DM 16 in the case of compact disks.

I grant herby special permission to distribute Fiasco on the "Fresh Fish" CD-Roms, on the "Meeting Pearls" CD-Roms and on the "Aminet" CD-Roms.

If you include Fiasco in your PD collection, coverdisk, etc. and a copy is left over, you may feel free to send me this copy.

1.3 Giftware

```
Giftware

Fiasco is Giftware , that means, that every User of Fiasco may
acknowledge the work I have done on Fiasco with some gift. This may be
money or any other little thing (CD-Roms, Books, etc.) or simply a
postcard (or nothing, if you think you could use your money for something
better; but please note, the ln_Pri of that is -128 ;-).

    My Address is:

    Nils Bandener

    Dekanatsgasse 4

    D-34369 Hofgeismar

    Germany
```

1.4 Filelist

```
Filelist

Fiasco Release 1.2 consists of these files:

Fiasco_1.2/ARexx.info
Fiasco_1.2/ARexx/age.rexx
Fiasco_1.2/ARexx/age.rexx.info
Fiasco_1.2/ARexx/arexxprint.rexx
Fiasco_1.2/ARexx/arexxprint.rexx.info
Fiasco_1.2/ARexx/graphprint.rexx
Fiasco_1.2/ARexx/graphprint.rexx.info
Fiasco_1.2/ARexx/print.rexx
Fiasco_1.2/ARexx/print.rexx.info
Fiasco_1.2/ARexx/unlockgui.rexx
Fiasco_1.2/ARexx/unlockgui.rexx.info
Fiasco_1.2/Catalogs/Deutsch/fiasco.catalog
Fiasco_1.2/Catalogs/Italiano/fiasco.catalog
Fiasco_1.2/Databases.info
Fiasco_1.2/Databases/Addresses.info
Fiasco_1.2/Databases/Addresses/Addresses.fdb
Fiasco_1.2/Databases/Addresses/Addresses.fdb.info
Fiasco_1.2/Databases/Addresses/Countries.fdb
Fiasco_1.2/Databases/Addresses/Countries.fdb.info
Fiasco_1.2/Databases/Addresses/Labels.fpr
Fiasco_1.2/Databases/Addresses/Labels.fpr.info
Fiasco_1.2/Databases/Addresses/ListLaTeX.fpr
Fiasco_1.2/Databases/Addresses/ListLaTeX.fpr.info
Fiasco_1.2/Databases/Addresses2.info
Fiasco_1.2/Databases/Addresses2/Adressen.fdb
Fiasco_1.2/Databases/Addresses2/Adressen.fdb.info
Fiasco_1.2/Databases/Addresses2/Adressmanager-Konv.rexx
Fiasco_1.2/Databases/Addresses2/Adressmanager-Konv.rexx.info
```

```
Fiasco_1.2/Databases/DatatypesDemo.info
Fiasco_1.2/Databases/DatatypesDemo/AmigaWorld.ilbm
Fiasco_1.2/Databases/DatatypesDemo/AmigaWorld.ilbm.info
Fiasco_1.2/Databases/DatatypesDemo/DatatypesDemo.fdb
Fiasco_1.2/Databases/DatatypesDemo/DatatypesDemo.fdb.info
Fiasco_1.2/Databases/DatatypesDemo/Hallelujah.8svx
Fiasco_1.2/Databases/FamilyTree.info
Fiasco_1.2/Databases/FamilyTree/families.fdb
Fiasco_1.2/Databases/FamilyTree/families.fdb.info
Fiasco_1.2/Databases/FamilyTree/persons.fdb
Fiasco_1.2/Databases/FamilyTree/persons.fdb.info
Fiasco_1.2/Databases/FAQs.info
Fiasco_1.2/Databases/FAQs/FAQS.fdb
Fiasco_1.2/Databases/FAQs/FAQS.fdb.info
Fiasco_1.2/Databases/FAQs/RunMost.rexx
Fiasco_1.2/Databases/FAQs/scantxtdir.rexx
Fiasco_1.2/Databases/FAQs/searchfaqs.rexx
Fiasco_1.2/Databases/FAQs/showtxt.rexx
Fiasco_1.2/Databases/GraphDemo.info
Fiasco_1.2/Databases/GraphDemo/Fragments.fdb
Fiasco_1.2/Databases/GraphDemo/Fragments.fdb.info
Fiasco_1.2/Databases/PD-Disks.info
Fiasco_1.2/Databases/PD-Disks/Disks.fdb
Fiasco_1.2/Databases/PD-Disks/Disks.fdb.info
Fiasco_1.2/Databases/PD-Disks/DisksLaTeX.fpr
Fiasco_1.2/Databases/PD-Disks/DisksLaTeX.fpr.info
Fiasco_1.2/Databases/PD-Disks/ReadFish.rexx
Fiasco_1.2/Databases/PD-Disks/ReadFish.rexx.info
Fiasco_1.2/Databases/PictureDatabase.info
Fiasco_1.2/Databases/PictureDatabase/Pictures.fdb
Fiasco_1.2/Databases/PictureDatabase/Pictures.fdb.info
Fiasco_1.2/Databases/PictureDatabase/scandir.rexx
Fiasco_1.2/Databases/PictureDatabase/showscr.rexx
Fiasco_1.2/Databases/Videos.info
Fiasco_1.2/Databases/Videos/CalcLen.rexx
Fiasco_1.2/Databases/Videos/Movies.fdb
Fiasco_1.2/Databases/Videos/Movies.fdb.info
Fiasco_1.2/Databases/Videos/Tapes.fdb
Fiasco_1.2/Databases/Videos/Tapes.fdb.info
Fiasco_1.2/Development.info
Fiasco_1.2/Development/fiasco.cd
Fiasco_1.2/Development/fiasco.cd.info
Fiasco_1.2/Development/fiasco.ct
Fiasco_1.2/Development/fiasco.ct.info
Fiasco_1.2/Development/Locale.readme
Fiasco_1.2/Development/Locale.readme.info
Fiasco_1.2/Documentation.info
Fiasco_1.2/Documentation/Deutsch.info
Fiasco_1.2/Documentation/Deutsch/fiasco.dvi
Fiasco_1.2/Documentation/Deutsch/Fiasco.dvi.info
Fiasco_1.2/Documentation/Deutsch/fiasco.guide
Fiasco_1.2/Documentation/Deutsch/Fiasco.guide.info
Fiasco_1.2/Documentation/English.info
Fiasco_1.2/Documentation/English/Fiasco.dvi
Fiasco_1.2/Documentation/English/Fiasco.dvi.info
Fiasco_1.2/Documentation/English/Fiasco.guide
Fiasco_1.2/Documentation/English/Fiasco.guide.info
```

```
Fiasco_1.2/Fiasco
Fiasco_1.2/Fiasco.info
Fiasco_1.2/gtlayout.library
Fiasco_1.2/icons/ARexx.info
Fiasco_1.2/icons/ARexxScript.info
Fiasco_1.2/icons/Databases.info
Fiasco_1.2/icons/def_FiascoPrint.info
Fiasco_1.2/icons/Documentation.info
Fiasco_1.2/icons/Drawer.info
Fiasco_1.2/icons/Fiasco.dvi.info
Fiasco_1.2/icons/Fiasco.guide.info
Fiasco_1.2/icons/Fiasco.info
Fiasco_1.2/icons/FiascoProject.info
Fiasco_1.2/icons/XPort.info
Fiasco_1.2/icons/XPortData.info
Fiasco_1.2/Install.info
Fiasco_1.2/Install/Deutsch.info
Fiasco_1.2/Install/English.info
Fiasco_1.2/Install/Install
Fiasco_1.2/Libs/MC68020.info
Fiasco_1.2/Libs/MC68020/gtlayout.library
Fiasco_1.2/XPort.info
Fiasco_1.2/XPort/mpearls_III_findpeals.fxp
Fiasco_1.2/XPort/mpearls_III_findpeals.fxp.info
Fiasco_1.2/XPort/RFF.fxp
Fiasco_1.2/XPort/RFF.fxp.info
Fiasco_1.2/XPort/StdTwist.fxp
Fiasco_1.2/XPort/StdTwist.fxp.info
```

1.5 Introduction

```
                Introduction

Fiasco is a Database for the Amiga. I originally wanted to write a simple
Program that could test one's English or Latin vocabulary. I later
implemented the ability to define more than two fields (answer and
question). The program continued to developed and finally became very
similar to a database program. I only needed to make minor changes and
there it was! Fiasco is now powerful, many featured program.

    Basically there is little difference between Fiasco and other
database programs. Although Fiasco does not support hierarchical
structures  (as AmigaBase does), it does support relations . Fiasco also
has an ARexx interface that can be used to control Fiasco from other
Programs or for assigning ARexx scripts to fields within a Fiasco
database.

    Fiascos's "mask" is not defined by a graphic file -- it is created
using internal images and any non-proportional font. Fiasco provides a
numver of field types. My personal favorite is the datatypes fieldtype
which can be used to display graphics, animations, texts etc. directly in
the
                Mask
```

.

"
Lists
" are a second way to display data. A list is much like the mask fully configurable. However, you cannot use a list to modify data.

The
searchsystem
of Fiasco supports "blurred" search and patterns. A "blurred" search tests for similarity between entries rather than equality. Fiasco's "similarity" threshold can be easily adjusted.

In addition, there are sort-, filter- and count-functions, which are related to the search system.

1.6 Features

Features

Fiasco has the following capabilities:

- Several projects may be in RAM at the same time. The number of these projects is only limited by the available RAM.
- Masks can be used like any other GUI.
- Masks, lists and requesters are fully font sensitive.
- Many fieldtypes: String, Integer, Float, Cycle, Boolean, Slider, Date, Time, Extern and Datatypes.
- Datatypes fields can be used to display graphics etc. directly in the mask.
- ARexx interface for external control and scripts for fields.
- Freely configurable "usermenu", which can be used to invoke CLI and ARexx Programs.
- Searching allows "blurred" search and patterns.
- Very flexible list, which supports hiding and resizing of entries
- Easy relation handling
- Import and export of Databases
- Flexible print function

1.7 News

News

Features added in Fiasco 1.2:

- Flexible print function.
- The internal sort function can sort records in relation to several fields rather than one.
- Bars for designing mask.
- F_SetFieldCont and F_GetFieldCont now have a record argument.
- New ARexx command: F_RequestField.
- Can save status windows in project file and restore it after loading.

Bugs fixed in Fiasco 1.2:

- Fiasco 1.1 crashed with an address error on Amigas with 68000/68010 processors.
- If the datatypes attribute Display filename was inactive, some text fields could simply disappear.
- If a datatypes field was positioned by Fiasco in an negative region, its contents might appear anywhere in the window. For now, those datatypes fields are not displayed.
- Search, count, etc. did not work with date and time fields.
- List sometimes left graphical garbage.
- Import/View did not work properly.
- After adding a field to a database which alreay contains records, the new entries will contain the init cont. Fiasco 1.0/1.1 left these fields empty or zero.
- F_SetFieldCont did not work for slider fields.

Features added in Fiasco 1.1:

- You may convert the field type of a field. E.g. you may convert a string field to a cycle field, whose labels correspondent to the old string contents.
- Import/Export of data.
- Read Only-field attribute.
- Virtual fields

- Fiasco can be run on its own screen.
- Fiasco detects itself in the system and will not start itself twice.
- Less memory fragmentation by using pools.
- Records may be marked.
- Extern and Datatypes fields may be now edited using file requesters.
- Datatypes fields support "Save" and deferred loading
- Datatypes fields display their messages in themselves
- Changes in the range of the labels for a cycle field don't affect the real contents anymore
- Button fields
- Relations are much faster!
- The requesters are created using gtlayout.library by Olaf Barthel
- Old ARexx commands extended and new added.
- Fiasco can talk to you
- The string fields in the mask cycle after Enter
- If Amiga OS 3.0 is available, Fiasco increases the size of the file-buffer when it reads or saves a project.
- The ARexx interface examines the arguments using ReadArgs() of the dos.library
- ARexx-Debug displays more information and has a help-button
- Documentation in dvi format for printed manuals
- A editor may be called from fieldreqs to edit ARexx scripts.
- Better pattern matching
- Support of clipboard

Bugs fixed in Fiasco 1.1:

- Usermenu did not allow to move entries in the list.
- Didn't close workbench.library
- Listwindow did not update the up/down-scrollbar if records were added
- Did not use the information about the font for the mask in the settings file.

- F_GetFieldCont and F_SetFieldCont could not be used for Float, Date, Time, Extern and Datatypes fields.
- Sometimes incorrectoy erased the graphic of fields.
- Crashed if you select Mask-Mode twice.
- Open in the mask mode did not activate the field under the cursor
- If you activated a project in mask mode, the service window was not updated.
- The options requester did not update the service window.

1.8 Requirements

Requirements

The minimum requirements are an Amiga with OS 2.04 (37.175) and 1 MB RAM. Recommended configuration: Amiga with OS 3.x (39.x or higher), 68020 Processor, 2 MB RAM and a Hard Disk.

Features and required OS-Versions:
Localization: Amiga OS 2.1 (38.x)
Screenmode-Requester: Amiga OS 2.1 (38.x)
Online-Help: Amiga OS 3.0 (39.x) or amigaguide.library v34 from FD
Datatypes-Fields: Amiga OS 3.0 (39.x)
Faster project-loading: Amiga OS 3.0 (39.x)

Fiasco 1.1 uses the gtlayout.library by Olaf Barthel for its GUI. The library is included in the archive of Fiasco.

The memory pool functions of Amiga OS 3.0 and Amiga OS 3.1 do not free unused puddles until the pool is deleted. Use SetPatch 40.16 (already included in WB 40.42) to fix this. If you use Amiga OS 2.0 or Amiga OS 2.1 you do not have to worry about that.

1.9 Installation

Installation

If you have the Installer program from Commodore simply doubleclick on the install icon of your preferred language in the install drawer. You then will be given step-by-step instructions.

If you don‘t own the Commodore Installer, you may simply drag the Fiasco drawer somewhere you want. You may copy the catalogs to locale:catalogs, but they will work at this place, too. You may delete the unused languages in "Documentation" and drag the remaining files in

```
the parent drawers. The files in "Development" and "Install" are not
required for normal operation of Fiasco and may be deleted, too. With
this configuration, Fiasco will run. If you have a 68020 processor or
better, you should delete the file gtlayout.library in the main directory
of Fiasco. Then, you should copy the gtlayout.library from the directory
libs/68020 into the main directory. If you want to make the
gtlayout.library accessible for all programs, you should copy it into the
libs: directory.
```

1.10 Quick Start

```
                Quick Start

These are the most important things, which you have to know while working
with Fiasco:


·  The program may be started over the Program- or Projecticon

·  There are two working-modes: In the record mode you may edit records,
   search for them etc. The mask mode allows you to add or modify fields.
   You may control the modes using the menuitems
                Control/RecordMode
                 and

                Control/MaskMode
                ·  The
                service window
                 makes certain operations easier, especially if you
   are not familiar with menu shortcuts. You may open it over

                Control/ServiceWin
                . Attention: The functions of the gadgets differ in
   the different modes.

·  A
                list
                , which can be opened with
                Control/ListWin
                , may be changed by
   clicking in the titles of the list. Clicking one time activates the
   column. Using the menu List you can do several things with this
   column. If you click at the right border of a title, you can size the
   column. The other space can be used to drag the column to any other
   place in the list.

·  Certain project options may be changed with the menuitem

                Project/Options
                ·  If you have any problems, you may press the help key while  ↩
                   browsing
   through the menu.
```

1.11 Basic elements of a Database

Basic elements of a Database

Basically, most databases are analogous to a card file.

A Fiasco database project consists of two components: First, there is the data which is divided into records. Second, there is the mask which defines the structure of the data.

The following pages describe the basics of databases in general and the basic Fiasco-specific principles.

Records

Fields

Mask

List

1.12 Records

Records

Records are the file cards of a database. That means a record is a collection of several data items for one main item (e.g. for a person name, address, etc.). In Fiasco the
mask
is only able to display one record at a time. The
list
displays several records as lines.

1.13 Fields

Fields

Fields define what data may be stored. In Fiasco the fields are defined in the mask. Fields are the basic elements of the mask and the list.

Fiasco supports several types of fields. More information on the field types and their features are located in the
field types
chapter.

1.14 Mask

Mask

The mask is the way to display data, which Fiasco uses most of the time. A mask, in constarst to a
list
, can display only one record. The advantage of the mask is the clarity of the display. In the card file example, the mask defines the structure of the file cards.

The mask consists of fields of which there are several types and images.

If you use normal Amiga programs, you would call these fields "gadgets". Internally, Fiasco uses gadgets (from the gadtools.library) as fields.

Fiasco masks adjust automatically to any non-proportional font. Topaz and courier are examples of non-proportional fonts.

To create a mask in Fiasco you have to be in the mask mode. You may change the position of existing fields using the mouse or make other changes with the Field menu. More on this topic
here
.

1.15 List

List

Control/ListWindow
opens a window, which displays the records in a list. The records are represented by lines, while the fields of a record are represented by columns. The first line of a list shows the IDs of each field. If the window is not big enough to display the whole list you may use the scrollbars in the right and in the bottom borders of the window to scroll through the list. The line of the current record is marked using a backfill.

You can select records using the list. Simply click on the line of a record. Changes to the record can only made in the mask.

If a
Filter
is active a list displays only the matching records.

Marked records

are displayed with a thin backfill.

The layout of a list is normally done automatically. Positions and dimensions of the fields in the mask will be used to determine the dimensions in a list. However, you may change the position and the width of each column in the list. Click in the header line at the right corner of a column to change the width. One line appears, which shows the actual width of the column. Now you may drag the line using the mouse. The place where you drop the line (that means you release the mouse button), will be new right border of the column. Columns, which are overlapped by the column, will be shifted to the right.

The position of a column may be changed, too. Click over the middle of the column header; you now may drag the column in the list. The column will be inserted as near as possible to the place where you drop it.

You may hide columns entirely with the menuitem
List/Hide column
. The
columns may be revealed by using
List/Show column
.

List/Recalc list
calculates the positions and dimensions of all
columns again. You can compare it with Clean up of the Workbench.
Columns, which have been hidden, are kept hidden.

1.16 Stretching of the mask

Stretching of the mask

Normally, the Fields in a Fiasco mask are placed very close together. This is not very nice and all other "normal" GUIs leave a few pixels between the gadgets. It is possible to place one empty line between the fields, but this wastes quickly a lot of place. For this reason Fiasco makes it possible to leave a few pixels between the gadgets.

These values may be specified in the
options requester
under Stretch
X and Stretch Y.

The owl stretching (ehhhmm -- mask stretching %-) makes fields bigger than specified in the field requesters. This is evident in the lines, because most Fiasco fields only expand to this direction. String fields may be bigger than the number of chars they can hold. The biggest problem are text fields, because their width is normally the minimum required. Stretching makes them wider and the text has to be centered.

You should specify zero as X value to avoid these problems and use one column as a separator. In Y direction this value, 4 is the best value.

1.17 Editing Modes in Fiasco

Editing Modes in Fiasco

Fiasco divides it's operation into modes. If you want to make changes in the mask, you have to be in the mask mode. If you want to make changes in the records, you have to be in the record mode.

Record Mode

Mask Mode

1.18 Record Mode

Record Mode

You may add, delete or edit records in this mode. It may be activated with

Control/Record Mode

. When the record mode is active, the field type cycle gadget in the service window is not selectable and the status gadget displays normally the number of the active record and the number of all records (for instance: 78 / 92).

1.19 Mask Mode

Mask Mode

This mode give you the ability to edit the mask, that is, you may create new fields, delete some or change their position or attributes. Relations may also created and changed here. This mode may be activated with

Control/Mask Mode

. When the mask mode is active, the "tape deck" gadgets in the service window are ghosted and the status gadget displays normally the coordinates of the cursor in the mask (for instance: X: 10, Y: 5).

1.20 Creating and working with a Database

Creating and working with a Database

And now to actual use: If you want to create a database in Fiasco you will have to create the mask at first and then the records. Fiasco allows

you in most aspects to create a database in an intuitive way.

The following sections describe the creation of a simple database.

Creating the Mask

Creating and working with Records

Converting Fields

Using marks

1.21 Creating the Mask

Creating the Mask

You have to activate the mask mode before you can create a mask (

Control/MaskMode
), whereupon a cursor will appear in the mask. You can use the mouse or the cursor keys to choose the location of the next operation in the mask. Before creating a new field you have to choose the type of the new field. You can use either the Field/Type
menu
the lowest
gadget in the service window to choose the field type.

You then may use
Field/Add Field
to create a new field. At first, the

field requester
appears. The gadgets in the requester depend on the supported attributes of the active field type. They are described in the

type documentation
for each field. It is not sufficient to click on Ok without any other action; you must specify certain attributes, such as the ID. Fiasco won't close the requester, if it contains any invalid settings. The field will appear in the mask after you close the requester.

You may change all attributes later except the fieldtype (Fiasco provides another function to do that). A field's position may be changed by dragging it with the mouse. The field requester may be opened by double clicking on the field or by choosing
Field/Edit Field
. You should
take care if you want to change the field ID. Other Fiasco projects or ARexx scripts which try to access this field won't find it after the change. If you change the value max chars of string, extern or datatypes fields, you will be informed, whether you could loose data.

With

```
                Field/Remove Field
                 you are able to delete Fields. Attention: If

                Settings/Security-Requester
                 is not active, all Data in this Field will be
freed immediately. Any existing project data on disk will be also erased
when the project is saved.

    You may specify further parameters for the current project, such as

                mask stretching
                , name of the author, etc. in the
                options requester
                .


                Field/Edit Relations
                 works similar to
                Edit Field
                . With this menuitem
you are able to control
                relations
                 of this field.

    When you have completed the mask you may return to record mode. You
are now ready to create records.
```

1.22 Creating and working with Records

```
                Creating and working with Records

You may create records for storing data in any mask containing fields.
The simplest wys to create a record is to select
                Record/Add Record
                 or its
equivalent Add in the service window. This creates, as the name implies,
a record and activates it. The fields in the record will contain the
values that have been assigned in the mask mode. You may now activate a
field using the mouse and edit its contents.

    Record/Duplicate Record provides another way of creating records.
This function creates a record, which is an exact clone of the record,
which was previously active. All init cont-attributes will be ignored.

    If no longer need a record you may delete it using

                Record/Remove Record
                 or Delete in the service window. If you have
selected Settings/Security-Reqs, you will be asked for confirmation
before the record is deleted.

    You may use the menu, the service window, the cursor keys or a list
window to view the records you have created.  I believed that the use of
GUI is intuitive, therefore, I will only explain the cursor keys. The
```

up-key activates the previous record. The down-key activates the next record. The order correspondents to the concept of a list window. The cursor keys combined with the Ctrl key activate the first or the last record respectively.

1.23 Converting Fields

Converting Fields

As your project develops you may decide that you want to change one or more of the field types. For instance, the contents of a field may have developed in a direction other than the one you originally intended. In that case, the convert function will be useful. This function is also helpful if you have imported a file. After a file is imported all fields are string fields.

You must be in mask mode to open the convert requester. Activate the field you want to convert and select Field/Convert Field. The convert requester displays the ID of the field, the current field type and the field types to which this field may be converted. If you select Alternative format, the convert function may convert the data to an other, often more abstract format. Not all field types support this option. If you select the the new type and proceed with Ok, the field will be converted. Note that Fiasco will not warn about the possible loss of data. If the new field type requires additional attributes (e.g., the extern fieldtype needs a program), the fieldrequester will open. Other attributes will use default values. If you convert a field and then convert it back to its original type it won't retain the original attributes.

Information about the results of a field type conversion can be found in the
field documentation
. Text and button fields cannot be converted.
In other cases, converting from one field type to another does not make much sense (e.g., boolean to datatypes).

1.24 Using Marks

Using Marks

Marks can be useful in advanced database use. A mark is simply a record's flag that may be toggled on or off, that is, a record is either marked or unmarked. Marks could be simulated using boolean or other fields, but the marking feature of Fiasco provides some additional advantages over that approach. First of all, a marked record can be easily discovered in the list because it is displayed in a highlighted state. If a marked record is active an "M" will be displayed to the right of the service window's status gadget, therefore, marked records can only recognized in a mask if the service window is open.

```
    Marks can be set using
                Record/Mark Record
                 and cleared using

                Record/Unmark Record
                . Use
                Record/Unmark all Records
                 to clear all marks in
a project. To set all marks, use
                Record/Mark all Records
                . Use

                Record/Toggle all Marks
                 to unmark all marked records and to mark all
unmarked records.


                Filters
                 are related to the marks. Thus, Fiasco provides some
additional menuitems in the Compare menu.
                Compare/Mark
                 opens a search
requester that can be used to mark all records that match a given
pattern. To convert the marks into a filter use
                Compare/Marks to filter
                .
To convert a filter into marks, use
                Compare/Filter to marks
                . These
functions convert set marks to "unfiltered" records and "unfiltered"
records to set marks.

    Marks are saved in a Fiasco project file and thus kept after
reloading of the project.
```

1.25 Searching in a database

```
                Searching in a database

The GUI interface to the Fiasco search function is the
                search requester
                .
Use
                Compare/Find
                 to open the search requester. The search requester
allows you to select the field to search and pattern to use in that
search. The controls for "blurred" search are also located here.

    The search pattern is simply the text you are searching for and can
include pattern matching as explained below. When you search a boolean
field, TRUE correspondents to a selected field and FALSE to a unselected
field. Cycle fields take the number of the label (counting from zero) or
the real text of the label. Slider fields only allow the value. Extern
and datatypes fields can only be searched by filename. You cannot search
```

(with the builtin function) the contents of a file.

    The gadgets at the bottom of the requester start the search. The record will be displayed if a matching entry is found. You can use the menuitems

                Compare/Find next
                 and
                Compare/Find previous
                 to continue your
search.

                Patterns

                blurred Search

                Searching with ARexx

                Counting

                Replacing

                Filter

1.26 Patterns

                Patterns

In addition to plain text you may use pattern matching. String fields support the use of patterns similar to AmigaDOS (although not exactly AmigaDOS patterns because "blurred Search" is not compatible with them). ? is equal to one unknown character. ?iasco would match Aiasco, Biasco, Ciasco, 1iasco, etc. ???? would match entries, which are 4 chars long. #? stands for an unknown number of unknown characters. A#? would match for example Amiga, Africa, A or ABCD. ?#? searches for all non-empty entries. Similar to AmigaDOS, these characters may be "escaped", if you want to search for entries, which contain these special characters. You have to precede a pattern character with a ' to enable search to find it.

    You can use * instead of #? by activating it through the menuitem

                Settings/Use * as pattern
                .

    Integer and slider fields support the following patterns: >, <, >=, <= , !=. The argument has to be given after the pattern. > only search for numbers greater than x, >= only for numbers greater or equal x, < only for numbers less than x, <= only for numbers less or equal x. != searches only for numbers not equal x. There is not pattern like == (equal), because this is represented by the number itself.

    The patterns supported by one fieldtype are also documented in the

                fieldtypes documentation

A
summary of all patterns
is also available .

1.27 Blurred Search

Blurred Search

"Blurred" search allows you to search for entries that are similar to a pattern. This enables you to search for entries even if you don’t know the exact spelling. The tolerance of the function may be set by "factor". 0 matches only entries that are exactly equal. 100 matches nearly all entries.

The
count function
is very suitable for experiments with "blurred" search.

1.28 Searching with ARexx

Searching with ARexx

You can also use ARexx to search a database. The commands
F_FindNext
,

F_FindPrev
and
F_FindFirst
can be used for this purpose. These commands take the field and the pattern for searching as arguments.

In contrast to the GUI search function, the record won‘t be activated. Only the number of the record will be returned in Result. This number can be used with other ARexx commands, such as,
F_GotoRec
.

If you call F_FindFirst, F_FindNext or F_FindPrev without arguments, these commands will use the arguments, which have been previously used in the search requester. You may also set these values using
F_SetSearchPat
and
F_SetSearchField
.

See the documentation for
F_FindFirst
for an example on searching

with ARexx.

    You can use ARexx to create a search function that supports several fields.

1.29 Count

Count

Compare/Count
 opens a requester similar to the search requester. As in the search requester you have to specify pattern, field and tolerance. If you select Ok the matches will be counted. This way you can collect experiences with the
blurred search
.

1.30 Replace

Replace

Compare/Replace
 is the function that enables you to replace certain values with others. Patterns are also possible here, but only one value will be inserted. The Replacement gadget takes the value to be inserted. If you select the Confirm gagdet you will be asked if you really want to replace the value for each record. The record will be displayed while you are asked.

    Attention: You can quickly destroy important data with a bad pattern (for example: #?)!!!

1.31 Filter

Filter

Fiasco's filter allows you to display only those records that match a pattern. With
Compare/Filter
 you may open the
filter requester
 which has
the same structure as the search requester. If select Ok only those records that match with the specified patter will be displayed.

    You may browse through the records with
Record/Next

and

Record/Previous
. The list also displays only matching records. You may temporarily disable the filter using
Compare/Filter On?
.

If you create new records while a filter is active the records will be displayed whether or not they match the filter pattern. If you change the contents of a existing record it will be also displayed. If you want to update the filter, you have to call the filter requester and select Ok .

1.32 Alternative Data Mechanisms

Alternative Data Mechanisms

Normally Fiasco stores field data directly in the project file. However, storing certain kinds of data this way is a very inefficient use of disk space.

Fiasco provides two alternative mechanisms for storing data. Relational projects read the data from another project into their projects. Several projects may access these data. In contrast, virtual fields store their data nowhere! The data are calculated automatically while loading the project.

Please note, that these mechanisms only help to save disk space. In RAM, they require the same amount of memory as other fields do.

Relations

Virtual fields

1.33 Relations

Relations

Relations are fields that store their contents in another project file rather than in the project file of the relations. An additional field is required that contains a key used to identify the record from which the data should be taken.

This mechanism prevents the situation, that in many different projects the same data are stored; it therefore saves disk space. Furthermore, you only have to change the contents of one field in one of the projects -- all other corresponding fields will also recognize that change.

Fiasco currently supports 1:1 relations. Such a relation connects one field in a project with another field in another project. Additionally, Fiasco supports a relation type with the name Sum:N. This relation type reads data from one project, adds them and puts the result in one field. This is not really a relation, because the read data will not and cannot be written back.

1.34 Creating Relations

Creating Relations

To use relations in Fiasco you have to create a project, which will be the data source for another project. The source project hereafter will be called "there" and the project that will read from it will be called "here".

You have to create at least two fields in the "there" project, one for the data and one for the key. The field for the key should be an integer field. This is the fastest method. However, it is possible to use any other field type as a keys.

You may use the special field attribute gimme unique key, if you want to automatically get a key whenever you create a new record. Note that the key is only created when you create a new record. If you activate this attribute later the already existing entries will keep their old value. If you change the contents of such a field the change will occur without any checking.

It is up to you to choose the type of the second field. If you create fields, which store strings (string, extern and datatypes), you should remember the max chars value because you also have to use this same value in the second project.

If you want to see any consequences of activating the relation, you should create a few records with some content at first.

Now it is time to save the project and to create a new one.

The second project also must contain two fields that have to match in type and max chars if you use string, extern or datatypes. The key field should not use unique key, because you should freely decide which key you want to use.

Before you activate the relation the project should be saved in the directory in which the other project has been saved in order to be able

to use relative rather than absolute paths.

    Now you can open the
                relation requester
                 for the field that is not
supposed to contain the key (
                Field/Edit Relations
). The topmost cycle
gadget should display 1:1. For "real" relations, you should keep this
choice activated. To start, you should select the key "here" in the
listview in the upper left edge of the window. After that you should
select the other project with the file requester gadget at the bottom of
the requester. Now you can select the key and the real field "there".
Proceed with Ok. If everything works correctly the requester will be
closed and the relations will be loaded. Otherwise, a requester will
inform you of any failure.

    A
                relation checklist
                , which contains the information in a compressed
form, is also available.

1.35 Technical notes about Relations

Technical notes about Relations

Fiasco 1.1 has increased the speed of accessing relations greatly. This
has been accomplished with certain optimizations, the basic code has not
changed. The action that consumes the most time when relations are
accessed is the search for correct keys. You have to go through the whole
"there" file and compare the keys for each key in your project. Fiasco
1.1 improves access speed by caching entries which have been read.
However, this consumes a great deal of memory. In low memory situations
Fiasco will have to throw out some cached entries to recover some memory.
As a result, Fiasco has to access the disk again to read these entries --
so you’re back to square one -- the whole process is slow. You should
ensure an adequate amount of free memory to avoid this.

    The second method of improving the access speed is to remember the
keys that have no matching key in the "there" file. This is, of course,
only useful, if your project contains some "blind" keys.

    The third method exists only because the first method exists. The
problem with this method is big files. It uses an unsorted list in which
the entries are stored with their record number. To get a record, Fiasco
will have to go through the whole list and compare each record number
with the one it is searching for. If this list contains a large number of
records this operation will be very slow (Imagine: If you have 1000
Records here, and 1000 Records there, you will have to examine 1000x1000
(MxN) records in the worst case). Sorting or special search and optimize
methods don’t solve this problem, these methods have a very high overhead
and slow the whole process even more. Fiasco 1.1 just remembers the
address of the record at which it previously stopped searching and
continues searching at this record next time around. This increases the

speed with certain files in which the growth of the keys is roughly the same as in the "there" files. I think (hope) that most files are structured in this way. However, the worst case still lies at MxN plus a small overhead required by this handling. This information should give you a rough idea of why one file does not load it's relations as fast as another. Here is a short list of the factors which may slow down the loading:

- Low memory
- No "blind" keys; All keys have a matching key in the "there" file
- Bad ordering of the records

Note: If you try to load relations from a floppy disk drive, it will get extremely slow, because Fiasco seeks through the whole file.

1.36 Virtual Fields

Virtual Fields

The data of virtual fields are not saved on disk; their data are calculated while loading the project. If you want to make a field virtual you should activate the Virtual option in the field requester.

Fiasco uses the ARexx script of a field for calculating these data. The script will be called for each virtual field in each record.

The number of commands that you can call is limited because Fiasco is in a special state. Currently, you may only call these commands:

- F_GetFieldAttributes
- F_GetFieldCont
- F_IsMarked
- F_MarkRecord
- F_RequestChoice
- F_RequestField
- F_RequestFile
- F_RequestNumber
- F_RequestString
- F_SetFieldCont
- F_UnmarkRecord

· F_VirtualMode

If you call any of these commands, which refers to a record, the record currently in work will be the active one (which is the default if you omit the record argument). If you want to refer to other records you should be aware of the fact that you don't know which virtual fields have already been completed. However, it is guaranteed, that all normal values and all relations are Ok. If you use F_GetFieldCont and F_SetFieldCont you don't know whether other virtual fields in the current record have been completed.

The ARexx script of a virtual field will be also called, like all other fields, after the contents are changed by the user. To find out, whether you are in normal or virtual state, use
F_VirtualMode
.

1.37 Printing a Database

Printing a Database

You can create a print-out of a Fiasco database in several ways. The internal print function is the easiest to use. To increase the quality of the print-outs, you may combine TeX with the print function. If you want to create a print-out that can't be created with the print function, you may use an ARexx script.

Internal Print Function

Printing with TeX

Printing with ARexx

1.38 Internal Print Function

Internal Print Function

The menuitem Project/Print opens the print window of Fiasco. This window is similar to a Fiasco project window in mask mode. It contains elements which can be arranged with the mouse. In the final print-out all records will be laid out that way.

When you open the print window Fiasco tries to open a file containing the standard print mask for the project. The file name for such files is Project Name.fpr. Project Name is the file name of the project without .fdb. If the file is not found, Fiasco lays-out the print mask according to the real mask.

To print the database as a list, you should select the menuitem Project/Get from list. This will layout the print mask according to the

real list. You may use Project/Get from mask to get the mask layout.

You simply have to select Project/Print to print the project with this layout.

The Print Mask

Print Mask files

1.39 The Print Mask

The Print Mask

The print mask has three parts: The head, the body and the foot. The head will be printed before any other data. The body will be printed for each record. It may contain references to project fields. These references will be substituted by the field contents of the records while printing. The foot will be printed last.

The print window displays only one of these parts at a time. To change the displayed part, use the Control menu.

The print window can be handled much like the project window in mask mode. To create an element (comparable to fields in the project mask), select a type with Element/Type and select Element/Add or press Return. Depending on the type, a requester will appear which gives you some options for the fields. Fiasco supports three element types:

- Field
- Text
- Formfeed

Field elements are usable only in the print body. They can be used to display the field contents in the print-out. The requester for field elements contains gadgets to select the field, to set the width, to set print styles like bold, italic or underlined and to activate clipping. Clipping can be used to control whether or not an entry may get wider than the specified width. If clipping is active, every entry which is longer than the width will be clipped to fit in that width. If clipping is not active, the following entries will be shifted.

Text elements are similar to text fields in the mask. They serve to put static text in the print mask. They support print styles like bold, italic and underlined. Text elements are the most important elements in the head and foot parts.

Formfeed elements terminate the page. That means that the data after a formfeed will be printed on a new page. Formfeed elements have no editable options and thus no requester opens after adding such a element. Edit element also is not usable for these elements. Because of the special meaning, the width of formfeed elements is "infinite". Formfeed

elements appear as a horizontal line in the mask.

1.40 Print Mask Files

Print Mask Files

Project/Save and Project/Save as in the print window create files which contain the print mask structure. These files can be reopened to restore a particular structure. If you have deleted a field in the database or if you have changed its ID, the print mask file may contain references to "nothing". When you open it Fiasco will try to get these references back. Fiasco uses a requester for that purpose which shows the field ID that was not found and a list of all fields in the current project. If you select one and click on Ok the reference will be changed to the selected field. If you Cancel the requester the element will be deleted.

You can easily adopt print masks to other projects this way. Simply load the database, open the print window and load the print mask. Now you can change all elements to the matching fields in the new database.

Besides the layout print mask, files contain the settings made in the

print options requester
.

1.41 Printing with TeX

Printing with TeX

You can use TeX to create high-quality print-outs of Fiasco projects. TeX is a kind of programming language, originally developed by Donald E. Knuth, which can be used to create printed documents. PasTeX is a freely distributable TeX implementation for the Amiga that can be found on many PD sites.

The print function of Fiasco supports TeX using
ARexx
.

If you select Print with ARexx in the
print options requester
, the
function of the Print menuitem of the print window is changed: After creating the print-out, the ARexx script ARexx/ARexxPrint.rexx is called with the name of the created file as its argument. This script should call TeX to compile and print the file. Because of that you must not write the file to PRT:. You should set Print to in the print options requester to a temporary file, for example T:FiascoPrintOut.tex. The script should look like this:

```
/* ARexxPrint.rexx
 * For use with PasTeX
```

```
 */

/* Parse arguments
 */
Parse Arg File

Address Command

File = strip(File,,'"')

/*  Call virtex
 */
'virtex' '"' || File || '"'

/*  Create name of dvi file
 */
dotpos = lastpos(".", File)

if dotpos ~= 0 then
    DVIFile = substr(File, 1, dotpos-1) || ".dvi"
else
    DVIFile = File || ".dvi"

/*  Call dviprint
 */
'dviprint' '"' || DVIFile  || '"'

/* Delete temporary files
 */
call delete(file)
call delete(dvifile)
```

If you want to print with TeX you have to create the print mask in a TeX compatible manner. For instance, you have to include a text element with the text \documentstyle{article} or something similar in the header if you work with LaTeX. Furthermore, the file must not contain any control charaters. Thus, Style attributes and formfeed elements cannot be used. The Fiasco distribution contains several examples for this.

1.42 Printing with ARexx

```
                Printing with ARexx
```

"Printing with ARexx" is a very comprehensive topic. This section should give you a rough idea of what can be done and how.

```
    One way of printing with
                ARexx
                 has already been explained in the
```
section Printing with TeX. You may "misuse" ARexxPrint.rexx for purposes other than calling TeX. For example, you may use a script which parses the data for your own purposes or loads it into your word processing program.

If you want to create more complex print-outs, which cannot be created with Fiasco's internal print function, you have to create the print-out with ARexx alone. Such an ARexx script has to go through the whole database and get the data it needs with
F_GetFieldCont
. After that it may do with the data what it wants.

The Fiasco distribution contains a complex example for such a script. The script GraphPrint.rexx is located in the ARexx directory and can be used with the GraphDemo project. However, it can be used with any other project that contains the required data. The script reads data from the project and creates an x/y diagram of the data. It automatically adapts to different value ranges. The script uses LaTeX and the eepic extension for the print-out. That means that you have to run a special host program in the background while printing. Because the script performs many mathematical operations it uses the rexxmathlib.library, which is not included in the distribution.

To start GraphPrint.rexx, click on the Graphic button in the GraphDemo/Fragments project. To use the script with another project, simply activate the project in Fiasco and start the script from the Workbench or Shell. Several requesters will appear. You have to specify what fields you want to use. You may select whether you want to view or print the TeX file directly or to write it to a specified location. After that the advanced options menu appears. To modify nothing, simply click on Continue. Edit Scale Base allows you to specify a value which will be used by the script as a base value for the scale of one of the axises. For example, if you use 5 (which is the default) you will get a scale of 5, 10, 15, etc. If you use 2 you will get 2, 4, 6, etc. Edit Origin allows you to choose whether the diagram will begin at point (0;0) or at a point which is the best for the project.

1.43 Import and Export

Import and Export

The Import and Export functions of Fiasco provide the ability to load data from other database programs into Fiasco and to write data with Fiasco that may be read by other programs.

Such Import/Export-files contain ASCII data. The fields or records are marked with special characters that may be freely defined in the Import/Export function of Fiasco.

Beginners, please note: Some basic knowledge is required to be able to effectively use Fiasco's Import/Export function. If you are familiar with databases you can skip the following information. The section

Special characters
describes the special escape sequences used by Fiasco. Although other databases may use a similar scheme you should read this section carefully. The whole Import/Export function of Fiasco relies on these escape sequences.

1.44 Structure of Import/Export files

Structure of Import/Export files

The names used here refer to the gadget labels in the Import/Export requesters. Note that some marking characters may be empty. To use the file with Fiasco you have to define, at minimum, either Field Start/Field End or Field Separator and either Record Start/Record End or Record Separator. However, the import functions of other programs may get upset, although this structure is correct.

Record Start
Field Start
Field Data Contents of the field in ASCII format.
Field End
Field Separator Separates two fields, not used after the last field of a record.
...
Field Start
Field Data
Field End
Record End
Record Separator Separates two records, notused after the last record of a file.
...
Record Start
... (see above)
Record End
End of File
If you activate First Record contains IDs, the field IDs will be stored in the first record as if they were fields.

An Example of an Import/Export file
Record start and record end are empty. Record separator is a newline character. Field start and field end are double quotes. Field separator is a comma. The first record contains the IDs of the fields. Note the empty field in the last record.

```
"Name","FirstName","Rank","Current"
"Picard","Jean-Luc","Captain","U.S.S. Enterprise"
"Riker","William Thomas","Commander","U.S.S. Enterprise"
"Data","","Lieutenant Cmdr.","U.S.S. Enterprise"
```

1.45 How to Specify Special Characters

```
How to Specify Special Characters

You often cannot simply type the characters for marking fields and
records as plain text. For example, if you want to use the newline
character as a record separator, you cannot simply hit the Return key.
Instead, you have to type it in as an escape sequence. Fiasco supports
escape sequences similar to the escape sequences of the "C" programming
language. The escape sequences are introduced by a \. These are
supported:


 \n Newline-character, ASCII 10
\f Formfeed-character, ASCII 12
\r Return-character, ASCII 13
\t Horizontal tabulator, ASCII 9
\v Vertical tabulator, ASCII 11
\Number Character with specified ASCII code
\Char Character directly copied
The last option ( \ + Character ) makes it possible to use a character,
which is reserved for escape-sequences.

    In Import, you may also specify character-classes.  Character-classes
are introduced in Fiasco with an #. These are supported:


#p Printable character.
#a Printable ASCII-character. Without international chars
#c Control-character. Not printable
Export supports to insert some additional information in the export-file.
These commands are introduced with an %. These are supported:


%f ID of field
%r Number of record
```

1.46 Importing of Data

```
                Importing of Data

The import requester is the GUI interface for Fiasco's import function.
You can open it using Project/Import. The file you want to import must be
specified in File. After having done this you have to specify the
structure of the file in the requester.If you are importing a file into
Fiasco immediately after export it from another database and still know
the structure parameters you can simply copy them into Fiasco's import
requester. Otherwise you can display the contents of the file using the
View button at the right side of the filename. Fiasco will start either
"More" or "MultiView" to display the file. If the file has a standard
structure it should not be too difficult to recognize the parameters.

    Usually, Record Start and Record End are empty and Record Separator
```

is \n. Field Start and Field End are often empty or double quotes (").
Usual values for Field Separator are a comma (,) or a tabulator (\t).

Skip Lines defines the characters that introduce a comment at the beginning of a line. If present, specify the comment introducer here. This may also be used to skip any formatting information present in the file. Fiasco's import function does not use such information. You can use Start Skip to skip any initial comment or similar items in the file. Max. Fields can be used to specify a record end mark if neither Record Separator nor Record End can be used.

Activate First Record contains IDs if the first record of the input file consists of Field IDs rather than real data. If you activate this the IDs will be used by Fiasco either to create fields with these IDs or to use already existent fields.

The options Append new fields and Overwrite old project control, whether you want to update a project or you want to create a new one. If you want to create a new project, you should activate both options.
Updating projects using Import

If you want to continue using your current settings you may save them with the Save button. Settings may be reloaded with Load. Fiasco already comes with several settings to import data from various sources.

To start the import process, you just have to click on Ok.
Attention: If the input file is too big, or even if the structure parameters are defective, the system may run out of memory! Fiasco has no big problems, if it runs out of memory, but other programs may have problems. For this reason, you should be careful with unsaved data!

If everything went well, the import requester will close and the new project will be activated. You will first want to improve the formatting of the project using the mask mode. If you did not activate First Record contains IDs, you should change the field IDs according to the contents of the fields. In addition, you should create text fields to labek the existing fields. At this point you have a nicely formatted project. However, all fields are string fields. You should determine whether some fields may be integer, cycle or other field types. You may change the type of these fields with the Fiasco's
convert
function. In the example
used in
Structure of Import/Export files
, the rank field may be
converted to a cycle field.

If you have followed these steps the project should be saved under a appropriate name.

1.47 Exporting of Data

Exporting of Data

From Fiasco's viewpoint, exporting data is much less complicated than importing. Normally, you can use Fiasco's default parameters (No Record Start and Record End, a newline character for Record Separator, double quotes for Field Start and Field End and a comma for Field Separator). If you use these parameters, you must take care, that you data do not contain any double quotation marks. In addition, you have to be certain that the program you want to import the data supports these parameters.

If you select First Record contains IDs, Fiasco will create an additional record at the top of the file which contains the field IDs. The file will contain no other formatting information.

If you select Marked Records only, only the marked records will be written.

Click on Ok to start exporting.

1.48 Fieldtypes

Fieldtypes

Data are stored in fields. There are only two basic types: "string" and "number". All other types are modifications, more or less, of these types which make the work with the database easier.

Fiasco supports the following types:

String

Integer

Float

Boolean

Cycle

Slider

Date

Time

Extern

Datatypes

Text

Button

Bar

1.49 Standard Attributes

Standard Attributes

These attributes are normally supported by a field type:

ID: This string serves for identification of a field. It is displayed in mask mode in the fields, in the list header, in the search and related requesters and in the relation requester. You also have to use it in ARexx scripts, if you want to access a field from there. This string must be unique in the current project.

Width: defines the width of the field in the mask in characters. This value is also used as a default value for the width of a list column. However, this can be changed separately.

Init Cont/Use own value: you may specify a value here which will be used while creating a new record.

Init Cont/Use old value: If you create a new record the value which has been used in the old record will be used in the new record.

Script: You may specify a ARexx script here which will be called, when a new record is created, or the content of a field is changed. It is possible that init cont will not have the effect specified in the requester, if the script changes the contents of the field.

Read Only: The field content will be displayed in a recessed box which cannot be activated or edited.

Virtual: The value of the field is not saved on disk, but is recalculated every time the project is loaded. This is done using the init cont attributes and the ARexx script attribute. Please note that these fields occupy the same amount of RAM as other fields.

By using
mask stretching
it is possible that the attributes, which specify the dimensions of the field, will be slightly influenced.

1.50 String Fieldtype

String Fieldtype

A string field takes strings with a designated length.

```
New Attributes:

Max Chars:   determines, how many chars may be typed in this field. This
   attribute has direct effect on the size of the project file.



Search equivalent:
correspondents to the content.

Supported search patterns:
? = One unknown character.
#? = No or more unknown characters.


Conversion into a string field:
Any field can be converted without loss of data into a string field.
Alternative formats, if supported are specified in parentheses.
Additional notes:

Boolean - "Checked" is TRUE(1), otherwise FALSE(0)
Cycle - Label (label number) converted
Slider - Level converted
Date - Date in format "DD.MM.[YY]YY" converted
Time - Time in format "HH:MM[:SS]" converted
```

1.51 Integer Fieldtype

```
Integer Fieldtype

You may enter integer numbers in the range from -2,147,483,348 to
2,147,483,347 in an integer field.

New Attributes:

Max Chars:   determines the maximum length of a number in chars.

Init Cont/Gimme unique Key:   puts a number unique to this database in
   this field whenever a new record is created. This Attribute is
   mutually exclusive to use own value and use old value.



Search equivalent:
is equal with the field content.

Supported search patterns:
>- greater than
<- less than
>=- greater or equal
<=- less or equal
!=- not equal


Conversion into an integer field:
```

```
Integer fields only accept the numeric part of the source data. If the
source data begin with a non-numeric character the field will contain 0.
Additional notes:

Float - Integer part converted
Boolean - "Checked" gets 1, "Unchecked" gets 0
Cycle - Label number converted
Slider - Level converted
Date - First date element (Day) converted
Time - First time element (Hour) converted
```

1.52 Float Fieldtype

```
                Float Fieldtype

You may enter a real number in a float field.

New Attributes:

Precision:   Number of digits after the decimal point.



Search equivalent:
is equal to the field content

Conversion into a float field:
Float fields only accept the numeric part of the source data. If the
source data begin with a non-numeric character, the field will contain 0.
Additional notes:

Boolean - "Checked" gets 1.0, "Unchecked" gets 0.0
Cycle - Label number converted


    Note: The precision of the float field type is not very high. It is
recommended to use
                string
                 fields instead. ARexx is also able to execute
mathematical operations with string fields if they contain only numerical
characters.
```

1.53 Boolean Fieldtype

```
Boolean Fieldtype

A Boolean field can contain only one of two values: "True" or "False". It
appears in the mask as a "checkbox gadget".

Changed Attributes:

Width:    always 3
```

```
Search equivalent:
TRUE or 1 - checked field
FALSE or 0 - unchecked field


Conversion into a boolean field:
Boolean fields convert all non-0 numbers and TRUE into the checked state.
All other values will be converted to the unchecked state.

    Under Amiga OS 2.x this field can look a bit strange because the
images are not scalable. Starting with OS 3.0, the size of the field is
adjusted to the font size.
```

1.54 Cycle fieldtype

```
Cycle fieldtype

Cycle fields have several choices from a freely definable list, this
helps to save memory. There is a maximum of 65536 choices. (I hope that's
enough ;-) A cycle field appears in the mask as a "Cycle gadget" (as the
name implies).

New Attributes:

Labels:   A list of all choices. There must be at least one entry, two
   entries make it a cycle field.



Search equivalent:
the number of the label, counting from zero or the entry itself (enter
correctly!)

Conversion into a cycle field:
The values will be converted into labels. If there are equal values they
will get the same label. Data are not lost.
Additional notes:

Boolean  -  "Checked" becomes TRUE(1), otherwise FALSE(0)
```

1.55 Slider fieldtype

```
Slider fieldtype

A slider is related to a integer field. It can be used to display integer
numbers graphically. The numbers may range from -32,768 to 32,767 and may
be influenced by several attributes.

New Attributes:
```

Min. Value: defines the smallest value. It corresponds to the position of the "knob" at the left or at the upper end of the field.

Max. Value: defines the highest value. It corresponds to the position of the "knob" at the right or at the lower end of the field.

Format: is a format string in style of the "C" programming language.
The syntax: %[-][0][Field][.Maximum][l]Format

· -: The number is left aligned, the default is right aligned

· 0: The field is padded with zeroes. e.g.: 1 -> 001

· Field: The minimal field width

· Maximum: only for strings, no meaning here.

· l: Says that the number is 32 bit wide. This is here always the case.

· Format:
c - Char, the ASCII character for the number is displayed.
d - The number is displayed.
u - The unsigned number is displayed.
x - The number is displayed in hexadecimal format.
There are also the b and s control characters. These take addresses as arguments and produce only garbage in this case.

The formatting is done with the exec-function RawDoFmt().

MaxFormatLen the maximum length of the format. This region is in the width region. That means that a higher MaxFormatLen makes the field itself smaller.

Search equivalent:
The number itself.

Supported search patterns:
> - greater than
< - less than
>= - greater or equal
<= - less or equal
!= - not equal

Conversion into a slider field:
Slider fields only accept the numeric part of the source data. If the source data begin with a non-numeric character the field will contain 0. You should check the range attributes after converting -- they could influence the data.

1.56 Date fieldtype

```
Date fieldtype

You may enter a date in a date field.

New Attributes:

Init Cont/use current Date:   When a new record is created the current
   date is copied in this field.



Search equivalent:
is equal to the contents

Conversion into a date field:
Date fields require the data in the format DD.MM.[YYYY]. The single parts
must be numbers. If values are non numeric, the part will get "??".
Additional notes:

Integer  -  converted to first element (Day)
Float  -  integer part becomes day, fractional part Month.
Time  -  Hour becomes Day


    Currently, Fiasco only displays and reads the date in German format
(DD.MM.[YY]YY). No verification of the values is made, this makes values
like 65.20.3687 possible.
```

1.57 Time fieldtype

```
Time fieldtype

You may enter a time in a time field.

New Attributes:

Init Cont/use current Time:  the current time will be copied in this
   field when you create a new record.



Search equivalent:
is equal to the content.

Conversion into a time field:
Time fields require the data in the format HH:MM:SS. Every element must
be a number. If an element is non numeric, it will be 0.
Additional notes:
```

```
Integer - Converted to hour
Float - Integer part converted to hour
Date - Day becomes hour


    Currently, the time is only displayed with seconds (HH:MM:SS). AM and
PM are not supported. No verification of the values is done, that means
that values like 55:66:99 are possible.
```

1.58 Extern fieldtype

```
Extern fieldtype

A extern field takes a string (most often a filename) that will be used
on request as argument for a user defined program. This makes it possible
to define additional data for a record.

New Attributes:

Command:   is the name of a program, which is capable of using these
   data. The characters %s are replaced with the content of the field. If
   you don't use %s, no arguments will be submitted. (For example type:
   C:ED %s )

Stack:   defines the stack size for a command.

Max Chars:   defines the maximum length of a filename in chars. This
   attribute has direct effect on the size of the project file.

FileReq Gadget:   select this attribute to have an gadget at the left
   side of the field that opens a file requester to edit the content. Of
   course, this only makes sense if the contents are filenames.



Search equivalent:
is equal to the content.

Conversion into a extern field:
All fields can be converted without loss of data into an extern field.
However, you have to specify a program that can use these data.
Additional notes:

Boolean - "Checked" becomesTRUE(1), otherwise FALSE(0)
Cycle - Label (Label number) converted


    The programs will be called using the AmigaDOS function System(). A
console window will be opened for I/O operations.
```

1.59 Datatypes fieldtype

Datatypes fieldtype

A datatypes field is similar to an extern field. The difference is the use of the datatypes.library. This is the reason, why you can use these fields only with Amiga OS 3.0 or greater. The major advantage is that the data will be displayed directly in the mask. A datatypes field is universal usable and freely extensible. A "popup"-gadget at the lower left side of the field makes it possible to edit the contents using an file requester. If something goes wrong, the error will be displayed in the field.

New Attributes:

Max Chars: defines the maximal length of the filename. This attribute has direct effect on the size of the project file.

Scrollbars: Determines if scrollbars will be created at the bottom and at the right border of the field. Without a scrollbar you can only view the upper left of a file. (That is not completely true. Some datatypes scroll their display if you click in their area and drag the mouse in the direction of the hidden part. The picture datatype is one example.)

Save gadget: If you activate this option you will get a second button under the datatypes field. The button will be marked with an S. If you select the button a file requester will appear which lets you choose a file to which the data, which are currently displayed in the field, will be saved. The data will be written in IFF format.

Display filename: When this option is active the filename is displayed at the bottom of the field in a string gadget. If you deactivate this option you cannot edit the value of the field.

Border: If this option is active Fiasco will render a border around the field. Do not deactivate this option too often because there are no visual elements which mark the beginning and the end of the field.

Defer loading: If you activate this option, the file of the field will not be immediately loaded when the record is activated. Instead, the message "Deferred" will be displayed in the field. Only if you activate the string gadget and hit return the data will be loaded and displayed.

Immediate play: Select this option to start playing of the data immediately after activating the record. If you activate this option, Defer loading must not be active. Of course, this option is only effective, if the datatype supports playing. The animation and the sound datatypes are such datatypes.

Searchequivalent:
Is equal to the filename; You cannot search the content.

Conversion into a datatypes field:
All fields can be converted without loss of data into a datatypes field.

```
However, the datatypes system requires valid filenames.
Additional notes:

Boolean - "Checked" becomesTRUE(1), otherwise FALSE(0)
Cycle - Label (Label number) converted


    The AmigaGuide and the animation datatype seem to have some problems
with relatively small fields.

    AmigaGuide datatype leaves sometimes graphical trash after scrolling
the contents.

    The changing of records gets slower, because the data have to be
loaded each time. To avoid that use Defer loading.
```

1.60 Text fieldtype

```
                Text fieldtype

Text fields are not real fields; these fields only serve to put text in
the mask.

Supported Attributes:

Text:   Will be written in the mask.

Pen:   The color used to write the text. The Normal default is black and
   the Highlight default is white. The colors can be manipulated with the
   palette prefs editor.

Bold:   Makes the text bold.

Italics:   Makes the text italic.

Underlined:   Underlines the text.

No
                standard attributes
                 are supported!

search equivalent:
You cannot search for a text field

Conversion into a text field:
You cannot convert any other fieldtype into a text field.
```

1.61 Button fieldtype

```
                Button fieldtype
```

```
Button fields only serve to put a button in the mask for a user-definable
action and are not real fields.

Supported Attributes:

Text:   will be displayed in the button.

Type:  Use Type to choose whether the button will execute a CLI or an
   ARexx program. CLI programs may be normal programs, commands or
   scripts (with the "s" attribute). ARexx programs must be ARexx
   scripts.

Command:  Use Command to select the program that will be executed when
   the button is activated.

Stack:   You may specify the stack size for the program here. The default
   is 4096. The program to be activated will crash if the stack size you
   specify is too small.

Console Window:   lets you specify the I/O stream for the program. It may
   be a console-window (CON:), the printer (PRT:), a simple file, or, if
   you don't want any output NIL:.



    The button fieldtype only supports the width-
                standard attribute
                .

search equivalent:
You cannot search for a button field

Conversion into a button field:
You cannot convert another fieldtype into a button field.
```

1.62 Bar fieldtype

```
Bar fieldtype

Bar fields only serve to put a visible separation in the mask and are not
real fields.

Supported Attributes:

Width/Height:  The width or the height of the bar, depending on Freedom.

Freedom:  determines whether the bar is drawn in the mask horizontally or
   vertically.



    The button fieldtype supports no standard attributes.

search equivalent:
```

```
You cannot search for a bar field

Conversion into a bar field:
You cannot convert another fieldtype into a bar field.
```

1.63 Fiasco's Graphic User Interface

```
                Fiasco's Graphic User Interface

Fiasco initially opens with an empty window. You can use the pull down
menus to work in it. The people who don't like pull down menus may open
an additional window using
                Control/ServiceWindow
                . This window makes the
most important operations accessible via a mouse click. Keyboard
shortcuts are the third way to execute operations.


                Menus and shortcuts

                Service window

                Requester
                    The mouse can be used in the mask mode to move the cursor and ↩
                       to drag
fields. After a double click on a field, its
                field requester
                 will be
opened (like
                Field/Edit field
                ).
```

Fiasco supports Menu help. If you press the help key while you browse through the menus, a short description will be displayed in an AmigaGuide window (This feature requires amigaguide.library, which is part of the OS since release 3.0. If you use 2.0 or 2.1, you may get it from the PD).

The requesters used by Fiasco have a standard structure. The gadgets at the bottom are for responding. Normally, the left one is a positive response, while the right one is negative. The close gadget of the window is equivalent to a negative response. Nearly all gadgets in the requesters may be accessed using the keyboard. Use the Return key for a positive response and the Esc key for a negative response.

1.64 The Service Window

```
                The Service Window

The service window may be opened or closed with
                Control/ServiceWindow
                . If
```

```
you want Fiasco to open the service window on every program startup,
select the menuitem
                Settings/Auto-Open ServiceWin
                . Select

                Settings/Dynamic ServiceWin
                 if you want Fiasco to search for a free place
on the screen when Fiasco opens the window. Otherwise, the position of
the service window at the time of saving the settings is used.

    The service window contains these gadgets:


                 Add

                 Del

                |<

                 <

                >

                >|

                 <Filename>

                  <Status>

                <Fieldtype>
```

1.65 Add

```
                Add

If the current project is in record mode a new record will be created. If
mask mode is active a new field will be created.

    Equivalent to:

                Record/Add
                 in record mode
resp.

                Field/Add field
                 in mask mode.
```

1.66 Delete

```
                Delete
```

```
If the current project is in record mode, the current record will be
removed. If mask mode is active, the current field will be removed.
Attention: This will normally happen without any security request!

    Equivalent to:

                Record/Remove
                 in record move
resp.

                Fields/Remove Field
                 in mask mode.
```

1.67 First

```
                First

If the current project is in record mode, the first record will be
activated.

    Equivalent to:

                Record/First
```

1.68 Previous

```
                Previous

If the current project is in record mode the previous record will be
activated.

    Equivalent to:

                Records/Previous
```

1.69 Next

```
                Next

If the current project is in record mode the next record will be
activated.

    Equivalent to:

                Records/Next
```

1.70 Last

```
                Last

If the current project is in record mode the last record will be
activated.

    Equivalent to:

                Records/Last
```

1.71 Active project

```
Active project

The name of the current project is displayed here. If two projects only
differ in the path and not in the name, the same name will be displayed.

    You may activate another project by activating the window of a
project.
```

1.72 Status

```
                Status

Status information is displayed here.

    In the record mode:
number of active record/number of records
A
                Filter
                 may change these numbers.

    In the mask mode:
X: X position of cursor, Y: Y position of cursor
```

1.73 Fieldtype

```
Fieldtype

If you are in record mode you can select the fieldtype which will be used
for subsequent calls of Add Field. Equivalent to:
Fields/Field Type.
```

1.74 Menus

```
                Menus

Fiasco has these pull down menus:
(from left to the right; menus, which are marked with a '/', may be
activated or deactivated)

    Name          Keyboardshortcut
Project

                New
                 A N

                Erase
                 A Z

                Open...
                 A O

                Options...
                 A $

                Statistic...

                Reload Relations
                 A !

                Save
                 A S

                Save as...
                 A A
Import...           A I

                Export...
                 A E

                Print...
                 A P

                About...
                 A ?

                Quit
                 A Q

Record

                Add Record
                 A +

                Duplicate Record
                 A 2

                Delete Record
```

```
                A -

               Delete all Records
                A @

               Cut Record
                A X

               Copy Record
                A C

               Paste Record
                A V

               Previous
                Cursor Up

               Next
                Cursor Down

               First Record
                Ctrl Cursor Up

               Last Record
                Ctrl Cursort Down

               Goto...
                A G

               Mark Record
                A  .

               Unmark Record
                A  :

               Mark all Records
                A  ,

               Unmark all Records
                A  ;

               Toggle all Marks
               Field

               Field Type        »
                 String    Ctrl S
Integer   Ctrl I
Float   Ctrl F
Boolean   Ctrl B
Cycle   Ctrl C
Slider    Ctrl S
Date    Ctrl A
Time    Ctrl M
Extern    Ctrl E
Datatypes Ctrl D
Text    Ctrl T
Button    Ctrl U
```

```
                Add Field...
                 Enter

                Edit Field...
                 Enter

                Duplicate Field

                Remove Field
                 Del

                Edit Relation...
                 A &

                Remove Relation
                 A  0

                Convert Field...
                 A "

List

                Hide column
                 A [

                Show column...
                 A ]

                Show all colums

                Recalc List
                 A %

Compare

                Find...
                 A F

                Find next
                 A >

                Find previous
                 A <

                Replace...
                 A R

                Count...
                 A #

                Sort...
                 A =

                Edit Filter...
                 A ~
```

```
                / Use Filter?
                 A `

                Mark...
                 A  K

                Filter to Marks

                Marks to Filter
                Control

                / Record Mode
                 A D

                / Mask Mode
                 A M

                / ServiceWindow
                 A W

                / ListWindow
                 A L

                / ARexx-Debug
                 A B

Settings

                / Create Icons?

                / Create Backups?

                / Write Rels?

                / Update Rels?

                / Security-Reqs?

                / AutoOpen SerWin?

                / Dynamic SerWin?

                Display...

                Editor...

                Save Settings

                Save Settings as...

                Load Settings...
                User

                Edit...
                 A U
```

1.75 Project/New

```
                Project/New

Shortcut: A  N

Creates a new project with a mask window. It contains no records or
fields. You may create a new database or
                Open
                 a saved database.

See also:
                Open
```

1.76 Project/Erase

```
                Project/Erase

Shortcut: A  Z

Erases all data in the current project. The project will be in a status
like immediately after calling
                Project/New
                . If data have been changed
since last saving, you will be asked before the data is erased.
```

1.77 Project/Open...

```
Project/Open...

Shortcut: A  O

Opens a file requester and loads the selected Fiasco project into the
current project window. If there are any unsaved data you will be asked
whether you want to save them first.

    If Amiga OS 3.0 is available, Fiasco will increase the buffer size of
the file. This speeds up the load process considerabily.
```

1.78 Project/Options...

```
                Project/Options...

Shortcut: A  $
```

```
This menuitem opens the
                options requester
                , which can be used for editing
project specific options. That are:

.
                Mask stretching
                ·  Name of author and annotations

·  filename of project

·  project windows

The point "filename of project" makes it possible to change to name
of the project without the need to call
                Save as
                .
```

1.79 Project/Statistic...

```
Project/Statistic...

no Shortcut

Shows some information for the current project. Example:
One Record requires about 100 Byte of RAM. 200 Records of this project
need about 19 KByte of RAM. There is space for about 2300 more Records.

    The memory required for the project and the basic fields is not
included.
```

1.80 Project/Reload Rels

```
                Project/Reload Rels

Shortcut: A  !

This item reloads all relations in the current project, just as they were
loaded while opening the project. This is particularly useful if you have
deactivated
                Settings/Update Relations?
                , changed some keys and want to see
the result.
```

1.81 Project/Save

```
                Project/Save

Shortcut: A  S
```

Save writes the data of the current project under the same name to disk.
If you want to save the project under a different name you have to use

Save as
or
Options
to change the name and then Save.

If Amiga OS 3.0 is available, Fiasco will increase the buffer-size of the file. This speeds up the save process considerably.

1.82 Project/Save As...

Project/Save As...

Shortcut: A A

You may save the current project under a new name here. The name will be requested using a file requester and will be kept after saving.

If Amiga OS 3.0 is available, Fiasco will increase the buffer-size of the file. This speeds up the save process considerably.

1.83 Project/Import...

Project/Import...

Shortcut: A I

Opens the
import requester
, the GUI interface for the import function of Fiasco. You can use import to load data from foreign databases into Fiasco.

1.84 Project/Export...

Project/Export...

Shortcut: A E

Opens the
export requester
, the GUI interface for the export function of Fiasco. You can use export to save data in a format which can be read by other databases.

1.85 Project/Print...

```
                Project/Print...

Shortcut: A  P

Opens the print window, the main interface to Fiasco's
                print function
                .
You may create a layout for printing here and print it.
```

1.86 Project/About...

```
Project/About...

Shortcut: A  ?

This item shows a small requester that displays informations about
version, copyright and some system internal data.
```

1.87 Project/Quit

```
Project/Quit

Shortcut: A  Q

This item closes the current project. If it has been changed and has not
been saved yet, you will be asked if you want to do this. If this is the
last open Fiasco project, Fiasco will exit.
```

1.88 Record/Add Record

```
                Record/Add Record

Shortcut: A  +

Adds a new record to the record list of the current project. Each Field
contains then its init cont, which is normally nothing. If the list is
open a new line will be inserted.

    If a
                Filter
                 is active the new record automatically will be declared
valid. If you want that new records are filtered correctly you will have
to select
                Compare/Edit Filter
                 again and simply click on Ok.

    This menuitem may only be selected in
```

```
                record mode
                .

See also:
                Record/Remove Record
```

1.89 Record/Duplicate Record

```
Record/Duplicate Record

Shortcut: A  2

Creates an exact copy of the current record. All init cont attributes
will be ignored. Even a field with gimme unique Key will contain the old
value. That means that two records with the same "unique" key will exist.
```

1.90 Record/Delete Record

```
                Record/Delete Record

Shortcut: A  -

Removes the current record and the data in it. If there are relations
which search for a key defined in this record, they will not find
anything in the future.

    This menuitem may be selected only in
                record mode
                . If you have
selected
                Setting/Security-Requester?
                , you will be queried before
proceeding.

See also:
                Record/Add Record
                ,
                Record/Delete all Records
```

1.91 Record/Delete all Records

```
                Record/Delete all Records

Shortcut: A  @

Removes all records in the current project. The mask will not be deleted
by this function.

    Record/Delete all Records may be only called in record mode.
```

```
    Note: Unlike the functions Delete Record and Remove Field, this
menuitem does not put up a security requester, if Security-Requesters is
activated. However, if the project has been changed, Fiasco will put up a
standard Ok-Save-Cancel-Requester.

See also:
                Record/Delete Record
```

1.92 Record/Cut Record

```
                Record/Cut Record

Shortcut: A  X

Copies the current record to the clipboard and removes it from the record
list of its project. After that, you may use
                Record/Paste Record
                 to
insert it in the project, again.

    This function may be called only in record mode.

See also:
                Record/Copy Record
                ,
                Record/Paste Record
                , Section Clipboard
support of Fiasco
```

1.93 Record/Copy Record

```
                Record/Copy Record

Shortcut: A  C

Copies the current record to the clipboard. You may use

                Record/Paste Record
                 to insert it in the project again.

    This function may be called only in record mode.

See also:
                Record/Cut Record
                ,
                Record/Paste Record
                , Section Clipboard
support of Fiasco
```

1.94 Record/Paste Record

```
                Record/Paste Record

Shortcut: A  P

Creates a new record and pastes the contents of the clipboard into that
record. Normally, you should call
                Record/Cut Record
                 or
                Record/Copy Record
                before calling this function.

    This function may only be called in record mode.

See also:
                Record/Cut Record
                ,
                Record/Copy Record
                , Section Clipboard
support of Fiasco
```

1.95 Record/Previous

```
                Record/Previous

Shortcut: Cursor up

Activates the record predecedin the current record. If the current record
is the first one, the display will be "beeped". Please note, that
                filters
                change the behavior of this item. In this case the previous  ↩
                   matching
record will be activated.

    The keyboard shortcut correspondents to the structure of the list
which displays the previous record over the current record.

    This menuitem may be only selected if
                record mode
                 is active.

See also:
                Next
                ,
                First
                ,
                Last
                ,
                Goto
                ,
                Find previous
```

1.96 Record/Next

```
                Record/Next

Shortcut: Cursor down

Activates the record after the current record. If the current record is
the last in the list, the display will be "beeped". Please note, that

                filters
                 change the behavior of this item. In the case of an active
filter, the next matching record will be searched.

    The keyboard shortcut correspondents to the structure of the list,
which displays the next record under the current record.

    This menuitem may be only selected if
                record mode
                 is active.

See also:
                Previous
                ,
                First
                ,
                Last
                ,
                Goto
                ,
                Find next
```

1.97 Record/First Record

```
                Record/First Record

Shortcut: Ctrl Cursor up

Activates the first record of the current project. In the case of an
active
                filter
                , for the first matching record will be searched.

    This item may be only selected in
                record mode
                .

See also:
                Next
                ,
                Previous
                ,
                Last
                ,
```

Goto

1.98 Record/Last Record

Record/Last Record

Shortcut: Ctrl Cursor down

Activates the last record of the current project. In the case of an active
filter
, for the last matching record will be searched.

This item may be only selected in
record mode
.

See also:
Next
,
Previous
,
First
,
Goto

1.99 Record/Goto...

Record/Goto...

Shortcut: A G

Opens the
goto requester
which can be used to activate a record using its number. Please note that the record number may be changed by adding or deleting records or by using
filters
.

This item can only be selected in
record mode
.

See also:
Next
,
Previous
,
First
,
Last

1.100 Record/Mark Record

```
                Record/Mark Record

Shortcut: A  .

Marks the current record. If a record is marked, it will displayed
highlighted in the list and the character "M" will be displayed in the
service window.

    This item can only be selected in
                record mode
                .

See also:
                Unmark Record
                ,
                Mark all Records
                ,
                Unmark all Records
```

1.101 Record/Unmark Record

```
                Record/Unmark Record

Shortcut: A  :

Deletes the mark of the current record. It won‘t be displayed highlighted
anymore.

    This item can only be selected in
                record mode
                .

See also:
                Mark Record
                ,
                Mark all Records
                ,
                Unmark all Records
```

1.102 Record/Mark all Records

```
                Record/Mark all Records

Shortcut: A  ,

Marks all records in the current project. Note that the previous marking
```

of all records will be overwritten.

    This item can only be selected in
                record mode
                .

See also:
                Mark Record
                ,
                Unmark Record
                ,
                Unmark all Records
                ,
                Toggle all Marks

1.103 Record/Unmark all Records

                Record/Unmark all Records

Shortcut: A ;

Clears the marks of all records in the current project. Note that the
previous marking of all records will be overwritten.

    This item can only be selected in
                record mode
                .

See also:
                Mark Record
                ,
                Unmark Record
                ,
                Mark all Records
                ,
                Toggle all Marks

1.104 Record/Toggle all Marks

                Record/Toggle all Marks

No Shortcut

Toggles the marks of all records in the current project. A marked record
will be unmarked and an unmarked will be marked. You can restore the
previous marking of the records by calling this menuitem once again.

    This item can only be selected in
                record mode
                .

```
See also:
                Mark Record
                ,
                Unmark Record
                ,
                Mark all Records
                ,
                Unmark all Records
```

1.105 Field/Fieldtype

```
                Field/Fieldtype

Select the current fieldtype in this submenu. It will be used if you
create fields. The cycle gadget in the
                service window
                 has the same
function.  These fieldtypes are available:



                String
                  Ctrl S

                Integer
                  Ctrl I

                Float
                  Ctrl F

                Boolean
                  Ctrl B

                Cycle
                  Ctrl C

                Slider
                  Ctrl S

                Date
                  Ctrl A

                Time
                  Ctrl M

                Extern
                  Ctrl E

                Datatypes
                  Ctrl D

                Text
                  Ctrl T
```

```
Button      Ctrl U
```

1.106 Field/Add Field...

```
                Field/Add Field...

Shortcut: Return

Opens the
                field requester
                 for the current field type and inserts the
created field at the current cursor position.

    This item can only be selected in
                mask mode
                .

    If there is already a field at the current cursor position nothing
will be done.

    Please note that Return is also shortcut for
                Edit Field
                . Return
creates a new field, if no field is currently active, otherwise, it opens
the requester for editing the active field.

See also:
                Edit Field
                ,
                Edit Relations
                ,
                Remove Field
```

1.107 Field/Edit Field...

```
                Field/Edit Field...

Shortcut: Return

Opens the
                field requester
                 for the selected field. The field requester can
be used to change several attributes of the field. If certain changes
would cause the lose of data (e.g. changing max chars of a string field
to a lower number), you will be informed about the problem and given the
opportunity to cancel the change. Field types may not be changed this
way. You have to use
                Convert Field
                .

    Please note that Return is also a shortcut for
                Add Field
```

```
                . Return
calls Add Field if no field is active and otherwise calls Edit Field.

    This item can only be selected in
                mask mode
                .

See also:
                Add Field
                ,
                Edit Relation
```

1.108 Field/Duplicate Field

```
Field/Duplicate Field

No shortcut
Makes an exact copy of the active field. It will be placed as near as
possible to the original field. The ID will be copy_of_FieldID.
```

1.109 Field/Remove Field

```
                Field/Remove Field

Shortcut: Del

Removes the selected field. All data in this field will be lost.
Relations or ARexx scripts which refer to this field will be not
functional. Attention: The relations or ARexx scripts will not complain
immediately after removing the field, but at the first activation.

    This item can only be selected in the
                mask mode
                .

See also:
                Edit Field
                ,
                Edit Relations
                ,
                Add Field
```

1.110 Field/Edit Relation...

```
                Field/Edit Relation...

Shortcut: A  &

This item opens the
                relation requester
```

```
                , which adds a
                relation
                 to the
current field.

    This item can only be selected in
                mask mode
                .

See also:
                Field/Remove Relation
```

1.111 Field/Remove Relation

```
                Field/Remove Relation

Shortcut: A  0

This item deletes all relation information for the active field. The data
in this field will be written into the normal file.

    This item can only be selected in
                mask mode
                .
```

1.112 Field/Convert Field...

```
                Field/Convert Field...

Shortcut: A  "

Opens the
                convert requester
                 for the selected field. Using convert you may
change the type of a field.

    This item can only be selected in
                mask mode
                .

See also:
                Add Field
                ,
                Edit Field
```

1.113 List/Hide column

```
                List/Hide column

Shortcut: A  [

Hides an activated column of the
                list
                . You activate a column by clicking
in the topmost line of the list which contains the field IDs. After
hiding a column, the columns at the right side of it will be shifted to
the left. The column may be made visible again by using
                Show column
                .

    This item may only be selected if the list window is open.
```

1.114 List/Show column...

```
                List/Show column...

Shortcut: A  ]

This item opens a requester which may be used to reveal the columns
hidden with
                Hide column
                . Fiasco tries to place the columns as near as
possible at their old positions.

    This item may only be selected if the list window is open.
```

1.115 List/Show all columns

```
                List/Show all columns

no shortcut

Makes all columns which have been hidden using
                Hide column
                , visible
again.

    This item may only be selected if the list window is open.
```

1.116 List/Recalc List

```
                List/Recalc List
```

```
Shortcut: A  %

This menuitem calculates all positions and dimensions of the columns in
the
                list
                . Hidden columns are not revealed.

    This item can be compared with Clean up of the Workbench.

    This item may only be selected if the list window is open.
```

1.117 Compare/Find...

```
                Compare/Find...

Shortcut: A  F

Opens the
                search requester
                 which can be used to define search criterions.

    This item is only selectable, if the
                record mode
                 is active and if the
current project contains at least one record.

See also:
                search requester
                ,
                Find next
                ,
                Find previous
```

1.118 Compare/Find next

```
                Compare/Find next

Shortcut: A  >

Activates the next record, which matches with the search criterions,
specified using the
                search requester
                . You will be informed if no matching
record is found.

    This item is only selectable if
                record mode
                 is active and if the
current project contains at least one record.

See also:
                Search requester
```

```
                ,
                Find
                ,
                Find previous
```

1.119 Compare/Find previous

```
                Compare/Find previous

Shortcut: A  <

Activates the previous record, which matches with the search criteria
specified with the
                search requester
                . You will be informed if no matching
record is found.

    This item is only selectable if
                record mode
                 is active and if the
current project contains at least one record.

See also:
                Search requester
                ,
                Find...
                ,
                Find next
```

1.120 Compare/Replace...

```
                Compare/Replace...

Shortcut: A  R

Opens the
                replace requester
                ,which can be used for replacing data.

    This item is only selectable if
                record mode
                 is active and if the
current project contains at least one record.
```

1.121 Compare/Count...

```
                Compare/Count...

Shortcut: A  #
```

```
Opens the
                count requester
                , which can be used to determine the number of
the records matching with the specified pattern.

    This item is only selectable if
                record mode
                 is active and if the
current project contains at least one record.

    See also:
                Find
```

1.122 Compare/Sort...

```
                Compare/Sort...

Shortcut: A  =

Opens the
                sort requester
                 which may be used to sort the records of the
current project.

    This item is only selectable if
                record mode
                 is active and if the
current project contains at least one record.
```

1.123 Compare/Edit Filter...

```
                Compare/Edit Filter...

Shortcut: A  ~

Opens the
                filter requester
                , which can be used to create
                filters
                .

    This item is only selectable if
                record mode
                 is active and if the
current project contains at least one record.
```

1.124 Compare/Use Filter?

```
                Compare/Use Filter?

Shortcut: A  ` (Gray key at the upper left of the keyboard)

This item can be used to switch the
                filter
                 on or off. If no filter has
been created yet the
                filter requester
                 will be opened.

    This item is only selectable if
                record mode
                 is active and if the
current project contains at least one record.
```

1.125 Compare/Mark...

```
                Compare/Mark...

Shortcut: A  K

Opens the
                mark requester
                 which can be used to mark specific records that
match a pattern. This works much like the creation of filters.

    Existing marks will be overwritten; marked records will be unmarked,
if they do not match.

    This item is only selectable if
                record mode
                 is active and if the
current project contains at least one record.
```

1.126 Compare/Filter to Marks

```
                Compare/Filter to Marks

No Shortcut

Converts the current
                filter
                 (active or inactive) to marks. Records that
match the filter will be marked and records that don't match the filter
will be not marked. If the filter is active, it will be deactivated.

See also:
                Compare/Edit Filter
```

,
Compare/Marks to Filter

1.127 Compare/Marks to Filter

Compare/Marks to Filter

No Shortcut

Converts the marking of the current project into a
filter
. Each marked
record will be declared as a valid record. Every record that is not
marked will be filtered out. This filter will not be copied into the
filter requester. If you open the filter requester and proceed with Ok,
the filter created with Marks to Filter will be overwritten.

See also:
Compare/Filter to Marks

1.128 Control/Record Mode

Control/Record Mode

Shortcut: A D

This item switches the current project to
record mode
in which records
and the contents of records can be changed. If this mode is active , a
checkmark will be set to the left side of the item.

See also:
record mode
,
mask mode

1.129 Control/Mask Mode

Control/Mask Mode

Shortcut: A M

This item switches the current project to
mask mode
in which the
mask
can
be changed. If this mode is active , a checkmark will be set to the left
side of the item.

```
See also:
                Mask mode
                ,
                Record mode
```

1.130 Control/ServiceWindow

```
                Control/ServiceWindow

Shortcut: A  W

This item controls the
                service window
                . If it is checked the service
window is open. The service window makes the most important record- and
mask-operations easier and displays some status information.

    The service window serves globally for all projects.
```

1.131 Control/ListWindow

```
                Control/ListWindow

Shortcut: A  L

This item controls the
                list window
                , if it is checked, the list is open.

    Each project may have its own list window.
```

1.132 Control/ARexx-Debug

```
                Control/ARexx-Debug

Shortcut: A  B

This activates a special debug mode of Fiasco for the
                ARexx interface
                . If
Fiasco commands fail, Fiasco will create a requester that contains more
detailed information about the error.
```

1.133 Settings/Create Icons?

```
Settings/Create Icons?

If this item is checked, Fiasco will create icons while saving projects.
```

1.134 Settings/Create Backups?

```
Settings/Create Backups?

This item determines, whether Fiasco creates backups of old projects
while saving new projects. The backup file will be named oldname.bak.
```

1.135 Settings/Write Relations?

```
                Settings/Write Relations?

If this item is checked, Fiasco will also write relations back in their
"there" projects. Otherwise, changes made in these fields will be lost.
This item should be only active if
                Update Relations?
                 is also active, or
if you call
                Project/Reload Rels
                 before saving. Otherwise you risk
overwriting data in the "there" project by invalid data in some fields of
the "here" project.
```

1.136 Settings/Update Rels?

```
                Settings/Update Rels?

This item determines whether
                relations
                 are updated immediately after the
input of a new key. This requires disk accesses which may become annoying
if Fiasco has to read the data from a floppy disk. If you deactivate this
item, you should also deactivate
                Write relations?
                , because there may be
invalid data in the project which would be written into the "there" file.
If you want to see the changes, you can update the relations using

                Project/Reload Rels
                .
```

1.137 Settings/Use * as Pattern?

```
Settings/Use * as Pattern?

Activate this item to activate the support of the asterisk as a valid
search pattern. The * has then the same meaning as #?.
```

1.138 Settings/Security-Reqs?

```
Settings/Security-Reqs?

If this item is checked, Fiasco will warn you before deleting any fields
or records. This can prevent erroneous deleting and loss of data.
```

1.139 Settings/Auto-Open ServiceWin?

```
                Settings/Auto-Open ServiceWin?

If this item is checked the
                service window
                 will be opened automatically
when Fiasco is started.
```

1.140 Settings/Dynamic ServiceWin?

```
                Settings/Dynamic ServiceWin?

If this item is checked the
                service window
                 will be opened in a free area.
Otherwise, fixed coordinates will be used.
```

1.141 Settings/Talking?

```
Settings/Talking?

Check this item, if you want Fiasco to use the narrator.device to "speak"
certain messages.
```

1.142 Settings/Display...

```
                Settings/Display...

no shortcut

Opens the
                display requester
                 which can be used to specify display options
for Fiasco. You can select whether Fiasco will open its windows on a
public screen or on its own custom screen. Furthermore, you may select
fonts for the screen and the mask.

    The latter replaces the menuitem Settings/Choose Font of Fiasco 1.0.
```

1.143 Settings/Editor...

```
Settings/Editor...

no shortcut

Opens a requester that lets you specify a editor program which will be
called by Fiasco if you select the Edit Script button in the field
requesters.
```

1.144 Settings/Save Settings

```
Settings/Save Settings

Saves the current program settings in the files "env:fiasco.prefs" and
"envarc:fiasco.prefs". The settings "survive" rebooting.
```

1.145 Settings/Save Settings as...

```
Settings/Save Settings as...

Saves the settings in a file, which has been specified with a file
requester. If you save the file in "env:", the settings won't survive a
reboot. If you save them only in "envarc:", they will only become active
after rebooting because Fiasco searches for its current settings in
"env:" and nowhere else.
```

1.146 Settings/Load Settings...

```
                Settings/Load Settings...

Loads and uses a specified settings file. To use them also after reboots
you should select
```

Save Settings
 to write them to "env:" and "envarc:".

1.147 User/Edit...

User/Edit...

Shortcut: A U

Opens the
usermenu requester
 which can be used to define usermenus.

1.148 The Print Window

The Print Window

The print window can be opened with Project/Print. You can control
Fiasco's print function from there. More on the print function in the

print section
.

The print window contains these menus:

Menu Shortcut
Project

Erase
A Z

Open...
A O

Get from Mask
A M

Get from List
A L

Save
A S

Save as...
A A

Print
A P

Options...
A T

```
                Exit
                  A Q

Element

                Element Type   »
                        Field      Ctrl F
        Text       Ctrl T
        Formfeed   Ctrl O

                Add...
                  Return

                Edit...
                  Return

                Duplicate
                Remove            Del

Control

                Edit Head
                  A H

                Edit Body
                  A B

                Edit Foot
                  A F
```

1.149 Project/Erase

```
Project/Erase

Shortcut: A Z

Removes all elements from the print window. After using this menuitem the
print window will be empty.
```

1.150 Project/Open...

```
Project/Open...

Shortcut: A O

Opens an file requester and reads the print layout from the selected
file. The old data will be overwritten.
```

1.151 Project/Get from Mask

```
Project/Get from Mask

Shortcut: A M

This menuitem tries to fake the project mask's layout in the print
window. The old print mask will be overwritten.
```

1.152 Project/Get from List

```
Project/Get from List

Shortcut: A L

This menuitem tries to fake the list's layout in the print window. The
old print mask will be overwritten.
```

1.153 Project/Save

```
Project/Save

Shortcut: A S

Select Save if you want to write the current print mask to a file on
disk. This is the file Project_Name.fpr, if you haven't selected another
using Open or Save as. The file name is displayed in the window title bar
of the print window.
```

1.154 Project/Save as...

```
Project/Save as...

Shortuct: A A

Select this menuitem if you want to save the print mask in another file
as the currently selected. The file name is displayed in the window title
bar of the print window.
```

1.155 Project/Print

```
                Project/Print

Shortcut: A P

This menuitem creates the print-out of the project using the active print
mask. The exact function of this menuitem is dependent on the settings
```

made in the
print options requester
.

1.156 Project/Options...

Project/Options...

Shortcut: A T

This menuitem opens the
print options requester
. Some print mask-specific
options may be edited here.

1.157 Project/Exit

Project/Exit

Shortcut: A Q

This menuitem closes the print window. The active print mask will be
deleted from memory.

You may also use the window's close gadget for this purpose.

1.158 Element/Element Type

Element/Element Type

Use this submenu to select the active element type. This type will be
used by the subsequent
Element/Add
calls.

These element types may be selected:

· Field (Ctrl F)

· Text (Ctrl T)

· Formfeed (Ctrl O)

More information can be found in the
print chapter
.

1.159 Element/Add...

```
                Element/Add...

Shortcut: Return

Creates a new element at the cursor position. The element will be of the
type set in the
                Element/Element Type
                 submenu. If the element type
supports a requester, the
                element requester
                 will appear.

    More information can be found in the
                print chapter
                .

    Note that this menuitem has the same shortcut as
                Element/Edit
                . This
shortcut will Add if no element is active and Edit if an element is
active.
```

1.160 Element/Edit...

```
                Element/Edit...

Shortcut: Return

Opens the
                element requester
                 for the active element. Elements can be made
active using the mouse or the cursor keys.

    Note that this menuitem has the same shortcut as
                Element/Add
                . This
shortcut will Add if no element is active and Edit if an element is
active.
```

1.161 Element/Duplicate

```
Element/Duplicate

No shortcut

Duplicates the active element.
```

1.162 Element/Remove

```
Element/Remove

Shortcut: Del

Deletes the active element. You may only recover this element using a
saved version of the print mask.
```

1.163 Control/Edit Head

```
                Control/Edit Head

Shortcut: A H

Selects the head part of the print mask for editing. The head part will
be printed before any other data. It may not contain field elements. This
menuitem, Edit Body and Edit Foot are mutually exclusive.

    See the
                print chapter
                 for more information.
```

1.164 Control/Edit Body

```
                Control/Edit Body

Shortcut: A B

Selects the body part of the print mask for editing. The body part will
be printed for each record. It may contain references to fields in the
form of field elements. These references will be substituted while
printing by the field contents. This menuitem, Edit Head and Edit Foot
are mutually exclusive.

    See the
                print chapter
                 for more information.
```

1.165 Control/Edit Foot

```
                Control/Edit Foot

Shortcut: A F

Selects the foot part of the print mask for editing. The foot part will
be printed after any other data. It may not contain field elements. This
```

menuitem, Edit Body and Edit Head are mutually exclusive.

    See the
                print chapter
                 for more information.

1.166 All Requesters

                All Requesters

Requesters are used by Fiasco to get information required for certain operations. Normally, the requesters are created after selecting Fiasco menuitem. So called EasyRequesters, which are used by Fiasco to request a simple choice are not explained here, because they are generally easy to understand and are described in function specific sections.

    Most requesters can be controlled by using the keyboard. The shortcuts, which are marked with an underscore, are usually single characters without a qualifier.

    The gadgets at the lower bottom of a requester are usually for proceeding. Normally, the left-most is a positive response (Ok), while the right-most is a negative response (Cancel). Return is the shortcut for the positive response. The gadget is additionally emphasized. Esc is the shortcut for the negative response.

                Edit Field

                Convert field

                Find

                Replace

                Count

                Sort

                Filter

                Mark

                Usermenu

                Project Options

                Goto

                Edit Relation

                Show column

                Display Options

Import

Export

Print Options

Print Element

1.167 Field requester

Field requester

The field requester can be used to change the attributes of a field. Each fieldtype has a different field requester because the gadgets of the field requester represent the supported attributes of each fieldtype. The supported attributes are listed with the documentation of each field . The field requester will show up if you call Add Field , Edit Field or double-click on a field.

If you proceed with Ok, all values will be checked for validity. If one value cannot be used by Fiasco, a requester will explain the problem.

A small summary of the conditions: (presuming that these attributes exist)

- There must be an ID.
- MaxChars must be > 0.
- Width must be > 2.

If dimensionvalues cannot be used, because other fields are too near the field, another requester appears with Shift, Squeeze and Cancel gadgets. Cancel does nothing other than return to the field requester. Squeeze makes the field fit the selected space. Shift moves the field to the left to make it fit. It is not always possible to Shift.

If you change an already existing field that stores its contents in strings and supports MaxChars (currently String , Extern and Datatypes), an additional control is implemented. If you change MaxChars to a value which does not allow to keep all strings in their original length (that means, some strings are longer), you will be asked if you want to

truncate these strings or keep the old value.

1.168 Convert Field requester

Convert Field requester

The convert field requester can be used to change the type of one field
without the need of an ARexx script.

Field ID: This text gadget displays the ID of the field that will be
converted. Please check here to see if you have called Convert Field
for the correct field.

Old Type: Displays the current type of the field.

New Type: Select the new type of the field here. Please note that
conversions between certain field types may cause a lose of data.
Consult the
field documentation
for more information on this topic.

alternative format: Select this checkbox to active an output-format
which differs from the normal format. Please see the
field docs
whether this gadget has any effect and if so, what.

Ok: Starts the conversion and then closes the requester.

Cancel: Simply closes the requester.

1.169 Search requester

Search requester

Field: Select the field that will be searched here. The listview
displays only the IDs of real fields. Buttons, bars and text fields
are not displayed. Only one field can be selected.

Pattern: Enter here the pattern for which to search. It may be a simple
value or one with
search patterns
. This value will be also used in the
count and replace requesters.

Blurred search/Activated: If you want to use
``blurred search''
you
will have to activate this gadget.

```
Blurred search/factor:   You can control the tolerance of the "blurred
   search" here. 0 searches only for exactly matching entries, 100
   searches for almost all entries.

Next:  Initiates the search for the next matching entry and activates it.

First:   Searches for the first matching entry.

Previous:   Searches backwards for the next matching entry.

Cancel:   Closes the requester without any further action.
```

1.170 Replace requester

```
                Replace requester

If you already know the
                search requester
                , you should have no problems
with this one.

Field:   Select the field which will be searched here. The listview
   displays only the IDs of real fields. Buttons, bars and text fields
   are not displayed. Only one field can be selected.

Pattern:   Enter the pattern here for which to search. It may be a simple
   value or one with
                search patterns
                . This value will be also used in the
   count and search requesters.

Replacement:   Enter a value here that will be copied in the matching
   entries. No patterns are possible.

Confirm:   If you want to be asked for every replacing operation, you
   should select this gadget.

Blurred search/Activated:   If you want to use
                blurred search
                 you will
   have to activate this gadget.

Blurred search/factor:   You can control the tolerance of the blurred
   search here. 0 searches only for exactly matching entries, 100
   searches for almost all entries.
```

1.171 Count requester

Count requester

This requester lets you count records, which match with a patterns. More on counting here . You can open this requester using Compare/Count .

If you are familiar with the search requester , you should have no problems with this one.

Field: Select the field here that will be searched. The listview displays only the IDs of real fields. Buttons, bars and text fields are not displayed. Only one field can be selected.

Pattern: Enter the pattern here for which to search. It may be a simple value or one with search patterns . This value will be also used in the search and replace requesters.

Blurred search/Activated: If you want to use blurred search you will have to activate this gadget.

Blurred search/factor: You can control here the tolerance of the blurred search. 0 searches only for exactly matching entries, 100 searches for almost all entries.

Ok: proceeds and counts the matching records. The number will be displayed at the end.

Cancel: closes the requester without any further action.

1.172 Sort requester

Sort requester

You can sort a Fiasco project with the sort requester bases on fields. It can be opened using Compare/Sort .

Fields: A list of all fields of the active project is displayed here. Clicking on one field will put it in the Sort by list.

Sort by: This list displays the fields by which the project will be

sorted. The topmost field has the highest priority while sorting, i.e. most data will be sorted according to this field. If there are entries which contain equal data the following fields will be used. Use the Delete to remove a field from this list. The arrows can be used to move the field in the list.

Descending: Select this gadget to sort the data from high values to low values (i.e. Z, Y, X, ..., C, B, A)

Ok: begins with sorting. The previously active record will be kept active, but it is highly probable, that the number of the record will change.

Cancel: Closes the requester without any further action.

1.173 Filter requester

Filter requester

Filters
offer the possibility of creating an overview of a group of records. A filter creates the impression of a database that consists only of the matching records. A filter is not created during the normal program functions; it is created once, immediately after activating the filter with the requester. Therefore, records added to the project while a filter is active will be displayed regardless of their contents. The same rules for record changes. The filter requester may be reached through
Compare/Edit Filter
.

If are familiar with the
search requester
, you should have no problems with this one.

Field: Select the field that will be searched here. The listview displays only the IDs of real fields. Buttons, bars and text fields are not displayed. Only one field can be selected.

Pattern: Enter the pattern here for which to search. It may be a simple value or one with
search patterns
.

Blurred search/Activated: If you want to use
blurred search
you will have to activate this gadget.

Blurred search/factor: You can control the tolerance here of the

blurred search. 0 searches only for exactly matching entries, 100 searches for almost all entries.

Ok: creates the filter. The project will appear to consist only of matching records.

Cancel: closes the requester without any further action.

1.174 Mark requester

Mark requester

Fiasco's mark function makes it possible to distinguish specific records. The purpose of the mark requester is to mark all records that match a specific pattern. This requester is closely related to the filter and

search requesters
. The mark requester may be opened with
Compare/Mark
.

Field: Select the field here that will be searched. The listview displays only the IDs of real fields. Buttons, bars and text fields are not displayed. Only one field can be selected.

Pattern: Enter the pattern here for which to search. It may be a simple value or one with
search patterns
.

Blurred search/Activated: If you want to use
blurred search
you will
have to activate this gadget.

Blurred search/factor: You can control the tolerance here of the blurred search. 0 searches only for exactly matching entries, 100 searches for almost all entries.

Ok: marks the matching records. The old record marks will be lost!

Cancel: closes the requester without any further action.

1.175 Usermenu Requester

Usermenu Requester

Fiasco has the ability to create own menuitems and to put CLI programs or ARexx scripts behind them. The defined items also may be selected with the F-Keys as well as with the mouse. F1 to F10 correspond to the first ten items, Shift and F1 to F10 correspond to the items 11 to 20. If you want to define more than 20 items, you will have to select the additional items with the mouse. Furthermore, Intuition limits the number of definable items to 63.

The items may be saved using
Settings/Save Settings
.

Items: This is the list of all existing menu items. You can Add one item and Delete one. < and > serve to change the position of the item in the menu.

Type: allows you to indicate whether the item will call a program or an ARexx script.

Command: Specify the program or the ARexx script here that will be executed.

1.176 Project Options requester

Project Options requester

The option requester contains project-related settings. It may be opened using the menuitem
Project/Options
or the ARexx command
F_OptionsReq
.

Name: You can change the filename of the project here. The project will be saved using this name in future. After saving, all direction-relative operations will use the new directory.

Author: You can use this field to enter your own name! It will be stored at the beginning of the project file.

Annotations: Yet another gadget for free use. You may store any notes here, for example a version string (with $VER: at the beginning). It will be written to the project file just before the author's name.

Mask Stretching X / Y: These values are added to the width or height of the cursor. The effect of this operation is a stretching of the mask in X- or Y-direction. More on stretching
here
.

Windows/Open list on startup: Activate this gadget to instruct Fiasco to open the list window when this project is loaded.

```
Windows/List position fixed:  If this gadget is active, Fiasco remembers
   the position of the list window when saving the project. After that,
   the list window will open at this position.

Windows/Mask position fixed:  If this gadget is active, Fiasco remembers
   the position of the mask window when saving the project. After that,
   the mask window will open at this position.
```

1.177 Goto requester

```
                Goto requester

The goto requester is one of the simplest of all the Fiasco requesters.
It may be opened with
                Record/Goto
                 and makes it possible to activate a
record using its number. Please note that
                Filters
                 change the record
numbers.


go to:   Takes the number of the record.

Ok:   Activates the record with the number.

Cancel:   Oh sorry, I just forgot... %-)
```

1.178 Relation requester

```
                Relation requester

This is the main interface for
                relation handling
                 in Fiasco. It may be
opened using
                Field/Edit Relations
                .

Type:  You have the choices 1:1 and Sum:N here. 1:1 is the standard
   relation. It searches for one correct key and copies the selected data
   in this project. Sum:N also searches for correct keys in the "there"
   project but adds the found data and copies the result in the field
   selected under Real here.

Key here:   Select the key in the current project.
```

Real here: Displays the ID of the field, whose relations are just now
   edited.

Key there: Use this listview to select the key-field in the project that
   has been specified under Related File. This listview displays only
   fields that can contain the key (with same type).

Real there: Use this listview to select the field of the project that
   has been specified under Related File and which is supposed to be the
   counterpart of Real here. This field is used to read the Data, which
   will be displayed in Real here. This listview only displays also
   fields that look as if they could contain the data (type and max chars
   must be equal).

Related File: Select the project file here relative to the directory of
   the current project which contains the informations.

Ok: Loads the relations. If any errors occur while loading, the
   requester will be activated again, otherwise it will return to the
   main window.

Cancel: closes the requester without any further action.

1.179 Show column requester

Show column requester

This requester, which may be reached with
List/Show column
, displays the
currently hidden columns in the
list
. If you select a column and click on
Ok, that column will be inserted in the list at its old position.

Field: All hidden columns are displayed here. Select the column here
   that you want to be revealed.

Ok: Inserts the column and re-displays the list.

Cancel: Closes the requester.

1.180 Display Options Requester

Display Options Requester

This requester controls the display elements of Fiasco. You can open your
own screen for Fiasco and choose the fonts for the custom screen and for

```
the mask (Fiasco 1.0 had the menuitem Choose Font for this purpose).

Screen

Screentype:  Select here, whether you want to use a public screen or an
   own custom screen.

PubScreen Name:  Specify here the name of the public screen, you want
   Fiasco to open its windows on. This has only effect, if you select
   PublicScreen for Screentype. If you leave this gadget empty, Fiasco
   will use the default public screen.

Screen Mode:  You may select the display mode here for the custom screen.
   Clicking on the popup gadget will open an ASL screenmode requester.
   This requires asl.library version 38 or higher.

Screen Font:  This gadget controls whether you want to use a custom font
   for the custom screen or the Workbench screen font which is controlled
   by the Font Preferences.

Custom Font:  If you want to use a custom font for the custom screen, you
   may select it here.

Mask Font

Mask Font:  This gadget controls, whether you want to use a custom font
   for the mask or the system default font which is controlled by the
   Font Preferences.

Custom Font:  You may select a custom font for the mask here. It must be
   fixed width.

Ok:  Proceeds with resetting the display of Fiasco. If required, a new
   screen is opened and so on.

Cancel:  Closes the requester without any further action.
```

1.181 Import requester

```
                Import requester

The import requester is the GUI interface to the
                import function
                 of
Fiasco. Import allows Fiasco to read data from other database programs.
Usually this cannot be done directly, but the foreign database has to
"export" the data. You may specify various parameters for importing, so
you should be able to read nearly all import/export-formats into Fiasco.

    The Fiasco distribution contains several predefined import formats,
which can be loaded using the Load button at the bottom of the import
requester.

    The values that may be typed in the gadgets of the import requester
```

are described in the Import/Export section of this document.

File: Specify here the file that contains the data to import. You may use the picker button at the right side to select it using a file requester.

View: Click here, if you want to view the contents of the file. Fiasco will start asynchronously More or MultiView, if available.

Records/Start: Enter the start characters for records here. Default: Empty.

Records/End: Enter the characters at the end of a record here. Default: Empty.

Records/Separator: Enter the characters between two records here. Default: \n.

Fields/Start: Enter the characters here that fields start with. Default: ".

Fields/End: Enter the characters here that fields end with. Default: ".

Fields/Separator: Enter the characters between two fields here. Default: \t.

Misc/Skip Lines: Enter introducing characters for remarks here. Default: Empty.

Misc/Start skip: Enter the number of lines here that will be skipped at the start. Default: 0.

Misc/Max fields: Enter the maximum number of fields in a record here. Can also be used if record separators are missing. Default: 100.

Options/First record contains IDs: Activate this gadget if the first record of the file contains the IDs of the fields in the project. They will be used by Fiasco then instead of generic IDs.

Options/Append new fields: Activate this, if you want Fiasco to create new fields for your data and not to use existing fields. If you have an entirely empty project you should activate this option.

Options/Overwrite old project: Removes the old data in the current project window. If you do not select this, you data will be appended in some manner to the existing project.

Ok: Starts the import process. Note that Fiasco may run out of memory due to bad structure parameters and overly large files. Programs that have problems with low memory should not run during this process.

Save: Saves the current settings in a specified file.

Load: Reads the settings from a specified file and sets them up in the requester.

```
Cancel:  Closes the requester without any further action.
```

1.182 Export requester

```
                Export requester

The export function provides the ability to share data from Fiasco with
other databases that cannot read the normal format of Fiasco databases.
See the
                Import/Export section
                 of this document for more information about
this mechanism.

File:  Specify the name of the file here that the data shall be written
   to. If a file already exists with this name it will be overwritten.

Records/Start:  Enter the start characters for records here. Default:
   Empty.

Records/End:  Enter the characters at the end of a record here. Default:
   Empty.

Records/Separator:  Enter the characters between two records here.
   Default: \n.

Fields/Start:  Enter the characters here that fields start with. Default:
   ".

Fields/End:  Enter the characters here that fields end with. Default: ".

Fields/Separator:  Enter the characters between two fields here. Default:
   \t.

Options/First record contains IDs:  Activate this gadget if you want
   Fiasco to write the field IDs in the first record.

Options/Marked records only:  Activate this gadget if you want Fiasco to
   write only marked records.

Ok:  Click here to start the export process.

Save:  Saves the structure parameters to a selected file.

Load:  Loads the structure parameters from a selected file.

Cancel:  Closes the requester without any further action.
```

1.183 Print Options Requester

Print Options Requester

The print options requester can be opened using Project/Options in the print window. You may control some options for printing here. See the Print section for more information on printing. The settings which have been made here may be saved using the menuitems Project/Save and Project/Save as of the print window.

Print to: Fiasco's print function writes its data to this file. If you want to use conventional printing, you should specify PRT: for the printer here.

Print with ARexx: Activate this gadget if you want Fiasco to call the ARexx script with the name ProgDir:ARexx/ARexxPrint.rexx after writing the file. Fiasco will call the script with the file name specified in Print to as its argument. In a standard Fiasco installation, this script calls TeX to compile the file into a DVI file and prints this. However, you may change the script to something completely different. If you use Print with ARexx, you must not specify PRT: in Print to. A temporary file, e.g. T:FiascoPrint, would be the best.

Only marked records: If you activate this gadget the print function will print only records with a mark.

1.184 Print Element Requester

Print Element Requester

You can control several options of a print element in the print window with this requester. It appears when you add with Element/Add or edit with Element/Edit an element. The layout of a requester depends on the in Element/Type selected element type. See the
print
section on more
information about elements.

1.185 ARexx

ARexx

ARexx is a macro programming language capable of connecting different programs. ARexx has been developed by William S. Hawes and is part of the system software since OS 2.0.

The ARexx port of Fiasco may be accessed externally from a script or ARexx scripts can be called by Fiasco. For example: you can specify an ARexx script in the
Script
field attribute and then change the contents

of a field -- these scripts may adjust the value of another field or do something else in response to the change.

To be able to communicate with Fiasco, you have to add the line Address FIASCO to the script.

Nearly all operations that can be used with Fiasco's GUI can be used with the ARexx commands. Additionally, Fiasco's functions may be extended with ARexx. There are many ARexx commands which do exactly the same as their GUI "brothers". In other words, certain commands may open a requester under certain conditions. It is often possible to circumvent this problem. It will be fixed sometime in the future. There are also commands which always open a requester. This also may be useful for scripts, but has been implemented primarily to give Fiasco a second menu (Iconbars). I have experimented with ToolManager-Docks. Unfortunately, it was too slow for fast browsing in a database.

General facts about using ARexx with Fiasco

Index of all ARexx commands

1.186 ARexx and Fiasco in general

ARexx and Fiasco in general

A Fiasco command returns in the case of success in RC 0. If a command had problems because its environment was not proper 5 is returned. More serious errors, like missing arguments, return 10. Fatal errors return 20.

Parameter are separated by white spaces. If single arguments are supposed to contain spaces, simply enclosing them in quotation-marks does not work. This is because ARexx swallows all quotation-marks. To avoid this you should enclose the marks in the other marks, e.g. F_Open ' "Test Datei" '). You have to use the single quotation marks in the outer position because Fiasco can only handle double quotation marks. Be sure not to use variables inside of any quotations. To use them, you have to close the quotation, write the variable and open the quotation again, if required. These issues do not apply for arguments which have the /F modifier.

If a command returns a value, this is stored in RESULT. To use RESULT , you have to put an OPTIONS RESULTS at the beginning of a script.

In an ARexx script started by Fiasco you cannot use path-searching with Address Command, because Workbench does not copy its path to the programs started by it.

The debugging of ARexx scripts is a bit problematic. Scripts which have been activated using the user menu or fields, have no output stream. All errormessages will be swallowed. If you want to test ARexx scripts, you should run the scripts from the shell (using rx filename; Fiasco must be in the correct status). To get more information about why a command that has been sent to Fiasco failed, you should activate the item

```
                Control/ARexx Debug
                . Fiasco will show a requester with an explanation for
the reason of the error. The script won't continue until the requester
has been closed. You have two choices there: Continue returns the correct
error code, Ignore Error returns 0 in RC, which looks like the command
has succeeded. An additional choice is Help, which won't proceed, but
displays the help text for the command, which failed.

    The style of the documentation of the commands is similar to the
Amiga OS Autodocs. Synopsis defines a template.


                Index of all ARexx commands
```

1.187 Index of all ARexx commands

```
                Index of all ARexx commands


F_AboutReq
 $^1$,$^2$

F_ActivateField

F_AddFieldReq
 $^1$,$^2$

F_AddRecord
 $^2$

F_ClearProject
 $^1$,$^2$

F_CloseServiceWin
 $^2$

F_CloseList
 $^2$~~~

F_ConvertField

F_CountRecs

F_CountReq
 $^1$,$^2$

F_DupRec
 $^2$

F_Export

F_FilterReq
 $^1$,$^2$
```

```
F_FindFirst
 $^2$

F_FindNext
 $^2$

F_FindPrev
 $^2$

F_FindReq
 $^1$,$^2$

F_GetFieldAttributes

F_GetFieldCont

F_GetProjName

F_GetProjFullName

F_GetRecNum

F_GotoFirstRec
 $^2$

F_GotoLastRec
 $^2$

F_GotoNextRec
 $^2$

F_GotoPrevRec
 $^2$

F_GotoRec
 $^2$

F_GotoRecReq
 $^1$,$^2$

F_Import

F_IsMarked

F_IsVirgin

F_LoadDTObject

F_Locate

F_LockGUI

F_MakeVirgin

F_MarkAllRecords
 $^2$
```

```
F_MarkMatch

F_MarkRecord
 $^2$

F_NewProject
 $^2$

F_OpenServiceWin
 $^2$

F_OpenList
 $^2$

F_OpenProject

F_OpenProjectReq
 $^1$,$^2$

F_OptionsReq
 $^1$,$^2$

F_Progress

F_Quit
 $^1$,$^2$

F_RemAllRecords
 $^1$,$^2$

F_RemRecord
 $^1$,$^2$

F_RequestChoice
 $^1$

F_RequestField
 $^1$

F_RequestFile
 $^1$

F_RequestNumber
 $^1$

F_RequestString
 $^1$

F_ResetStatus

F_SaveProject

F_SaveProjectReq
 $^1$,$^2$

F_SaveSettings
 $^2$
```

```
                F_SetFieldCont

                F_SetMode
                 $^2$

                F_SetSearchPat

                F_SetSearchField

                F_SetStatus

                F_Sort

                F_SortReq
                 $^1$,$^2$

                F_SelectProj

                F_ToggleAllMarks
                 $^2$

                F_UnlockGUI

                F_UnmarkAllRecords
                 $^2$

                F_UnmarkRecord
                 $^2$

                F_UserCommand

                F_VirtualMode
                $^1$ -- Commands, which may open an requester or something similar ↩
                    .
$^2$ -- Commands, which can be used to emulate menu-functions. Most commands
with an $^1$ have also a $^2$.
```

1.188 F_AboutReq

```
                F_AboutReq


Name:  F_AboutReq  -- Open the "About" requester

Synopsis:  F_AboutReq

Function:  Does exactly the same as
                Project/About
                .

Inputs:  none

Results:  none
```

1.189 F_ActivateField

```
F_ActivateField


Name:   F_ActivateField -- activate the field in the GUI

Synopsis:   F_ActivateField Field/A
   rc = Success

Function:   Activates the field with the specified ID in the mask. Only
   fields, which appear as a string/longint gadget may be activated. If
   project or window is not active, the field cannot be activated. This
   command may be only called in record mode.

Inputs:   Field - ID of field to activate

Results:   rc = 0, if field has been activated

Bugs:  Because of the automatic activation of fields performed by Fiasco,
   it cannot be used in an ARexx script of a field. Fiasco activates the
   field before the script is able of doing that. If F_ActivateField is
   finally called, the other gadget is still active and Intuition refuses
   to activate the gadget.

See also:   intuition.library/ActivateGadget()
```

1.190 F_AddFieldReq

```
               F_AddFieldReq


Name:   F_AddFieldReq -- open the
               add field requester
               Synopsis:   F_AddFieldReq

Function:   This command does exactly the same as
               Field/Add Field
               .  It
   may be only called in
               mask mode
               .

Inputs:

Results:

See also:
```

1.191 F_AddRecord

```
                F_AddRecord


Name:   F_AddRecord -- Add a new record.

Synopsis:   F_AddRecord

Function:   Add to the current project a new record. This record will get
   active.  This function may only be called in record mode.

Inputs:   none

Results:   none

See also:
                F_RemRecord
                ,
                Record/Add Record
```

1.192 F_ClearProject

```
                F_ClearProject


Name:   F_ClearProject -- clear the active Project

Synopsis:   F_ClearProject Force/S

Function:   Deletes all data in the current project. It will be in a
   state much like after a New.  If you do not specify Force, this
   command does exactly the same as
                Project/Erase
                . That means, it is
   possible, that a requester opens, which asks you whether you want to
   save the current project before proceeding or cancel.  To prevent
   this, specify the Force parameter. This will suppress all warnings. To
   find out, wheter the project is not saved, you may use
                F_IsVirgin
                .

Inputs:   Force -- suppress all warnings

Results:   none

See also:   Project/Erase,
                F_IsVirgin
                ,
```

```
                F_MakeVirgin
```

1.193 F_CloseList

```
                F_CloseList


Name:   F_CloseList -- close the
                list window
                Synopsis:   F_CloseList

Function:   This command is equal to deactivating the menuitem

                Control/List
                .

Inputs:

Results:

See also:   Control/List
```

1.194 F_CloseServiceWin

```
                F_CloseServiceWin


Name:   F_CloseServiceWin -- close the service window

Synopsis:   F_CloseServiceWin

Function:   Closes the service window. If the window is not open, nothing
   happens.

Inputs:   none

Results:   none

See also:
                F_OpenServiceWin
```

1.195 F_ConvertField

```
F_ConvertField
```

```
Name:   F_ConvertField -- change the type of a field

Synopsis:   F_ConvertField Field/A,NewType/A,AltFormat/S

Function:   Changes the type of the named field. You cannot convert text
   or button fields. May be only called in mask mode.

Inputs:   Field - ID of field
   NewType - New Type of field. (e.g. String)
   AltFormat - Specify, if you want an alternative Format

Results:   none

See also:   Chapter Converting Fields
```

1.196 F_CountRecs

```
F_CountRecs


Name:   F_CountRecs -- count the records

Synopsis:   F_CountRecs
   Result = Number_of_Records

Function:   Counts the records, which are currently in the current
   project. May be only called in record mode.

Inputs:   none

Results:   Number_of_Records - The number of records, may be zero. Note,
   that Filter influence this value.

See also:
```

1.197 F_CountReq

```
                F_CountReq


Name:   F_CountReq -- Open the count requester

Synopsis:   F_CountReq

Function:   Does exactly the same as
                Compare/Count
                . May be only called in
   record mode.

Inputs:
```

```
Results:

See also:   Compare/Count...
```

1.198 F_DupRec

```
               F_DupRec


Name:   F_DupRec -- Clone the active record

Synopsis:   F_DupRec

Function:   This command duplicated the active record exactly. All the
   Init Cont attributes are ignored. This command does exactly the same
   as
               Record/Dup Record
               . May be only called in record mode.

Inputs:

Results:

See also:   Records/DupRecord
```

1.199 F_Export

```
               F_Export


Name:   F_Export -- export ata out of Fiasco

Synopsis:   F_Export File/A,RecStart/K,RecEnd/K,RecSep/K,FieldStart/K,
   FieldEnd/K,FieldSep/K,FirstRecIDs/K,MarkedOnly/S
   rc = Success


Function:   Calls the export function of Fiasco. See the Import/Export
   chapter for more information about exporting. If you do not specify a
   parameter, it will be empty.

Inputs:   File - File to write
   RecStart,RecEnd,RecSep,FieldStart,FieldEnd,FieldSep - structure
   parameters
   FirstRecIDs - First Record will contain field IDs
   MarkedOnly - Exports only marked records
```

```
Results:   rc = 0, if everything went well.

See also:
               F_Import
               , Chapter Import/Export
```

1.200 F_FilterReq

```
               F_FilterReq


Name:   F_FilterReq -- open the
               filter requester
               Synopsis:   F_FilterReq

Function:   Does exactly the same as
               Compare/Filter
               . May be only called
   in record mode.

Inputs:

Results:

See also:   Compare/Filter
```

1.201 F_FindFirst

```
               F_FindFirst


Name:   F_FindFirst -- Search for a pattern

Synopsis:   F_FindFirst Field,Blur/K,Pattern/F
   Result = Number_of_Record

Function:   Searches for the first matching with the pattern, which has
   been either set with
               F_SetSearchPat
                or using the arguments. If rc is
   equal zero, Result is equal to the number of the found record. This
   may be accessed using
               F_GotoRec
               . If nothing is found, 5 is returned.

Inputs:   Field - ID of the Field to search
   Blur - Factor for blurred search. Specifying activates it.
   Pattern - Standard search pattern.
```

```
   If you don't specify Field or Pattern, the values will be used, which
   have been previously used in the search requester or have been set by
   F_SetSearchPat and F_SetSearchField.

Results:   rc = 0: result = Number of matching record.  rc = 5: nothing
   found or no pattern.

Example:

/* Find-Example.rexx */
options results
address FIASCO

count = 0

F_FindFirst "Test" "?#?"   /* search for the first record in
                            * which the field with the ID Test
                            * is not empty */

do while rc = 0            /* Continue searching until
                            * nothing is found */

   F_GotoRec Result        /* activate the found record */

   count = count + 1

   F_FindNext "Test" "?#?" /* search for next */

end

/* All records done */




See also:
```

1.202 F_FindNext

```
               F_FindNext


Name:   F_FindNext -- Search for a pattern.

Synopsis:   F_FindNext Field,Blur/K,Pattern/F
   Result = Number_of_next_Record

Function:   Searches for the next matching with the pattern, which has
   been either set with
               F_SetSearchPat
                or using the arguments.  If it
   succeeds ( rc = 0), Result contains the number of the found record.
```

```
   The record may be activated using
                F_GotoRec
                .

Note:   The active Record is not searched by F_FindNext and F_FindPrev.
   If you want to write a program, which searches all records, you have
   to call at first
                F_FindFirst
                 and then F_FindNext.

Inputs:   Field - ID of the Field to search
   Blur - Factor for blurred search. Specifying activates it.
   Pattern - Standard search pattern.
   If you don't specify Field or Pattern, the values will be used, which
   have been previously used in the search requester or have been set by
   F_SetSearchPat and F_SetSearchField.

Results:   If rc = 0, result = recordnumber of next matching.
   If rc = 5, nothing found or no pattern

Example:   see
                F_FindFirst
                See also:
```

1.203 F_FindPrev

```
                F_FindPrev


Name:   F_FindPrev -- Search for a pattern backwards.

Synopsis:   F_FindPrev Field,Blur/K,Pattern/F
   Result = Number_of_prev_Record

Function:   Searches for the previous matching with the pattern, which
   has been either set with
                F_SetSearchPat
                 or using the arguments.  If
   it succeeds ( rc = 0), Result contains the number of the found record.
   This may be activated using
                F_GotoRec
                .

Inputs:   Field - ID of the Field to search
   Blur - Factor for blurred search. Specifying activates it.
   Pattern - Standard search pattern.
   If you don't specify Field or Pattern, the values will be used, which
   have been previously used in the search requester or have been set by
   F_SetSearchPat and F_SetSearchField.

Results:   if rc = 0, result contains the recordnumber of previous
   matching if rc = 5, nothing found or no pattern
```

```
Note:   F_FindPrev is not very handy in ARexx scripts. You should use
   combinations of
                F_FindFirst
                 and
                F_FindNext
                 instead.

See also:
```

1.204 F_FindReq

```
                F_FindReq


Name:   F_FindReq -- open the search requester

Synopsis:   F_FindReq

Function:   Opens the
                search requester
                . This command does exactly the
   same a
                Compare/Find
                . The command may be only called in
                record mode
                .

Inputs:   none

Results:   none

See also:
```

1.205 F_GetFieldAttributes

```
F_GetFieldAttributes


Name:   F_GetFieldAttributes -- Read the attributes of a field

Synopsis:   F_GetFieldAttributes Field/A,X/S,Y/S,W=Width/S,H=Height/S,
   Rexx/S,Type/S,ListX/S,ListW/S,MaxChars/S,InitCont/S,OwnInit/S,
   Labels/K/N,Commands/S,Stack/S
   rc = Success
   Result = Attribute_Value

Function:   Reads one attribute of the specified field. The value of the
   attribute is returned in Result. Not every fieldtype supports all
```

```
   attributes, if a type does not support a particular attribute, rc will
   be not equal 0. You may only specify one attribute while calling this
   command. For convenience, this command may be called both in record
   mode and in mask mode.  This command may be also called in virtual
   state.

Input:   Field - ID of a field. Always required.
   X - I want to know the top edge of field in cursors
   Y - Left edge of field in cursors
   W - Width of field in cursors
   H - Height of field in cursors
   Rexx - Name of ARexx script assigned to field
   Type - Type of field (e.g. string, integer, etc.)
   ListX - Left edge of field in list, -1 if field is hidden
   ListW - Width of field in list, -1 if field is hidden
   MaxChars - MaxChars attribute
   InitCont - InitCont attribute. One of own, old, key OwnInit - Own
   initial content
   Labels - Returns the label of the specified number
   Command - Command attribute of field
   Stack - Stack attribute of field


Results:   rc - zero, if successful.
   Result - contains requested attribute, if rc = 0

See also:   Field documentation
```

1.206 F_GetFieldCont

```
F_GetFieldCont


Name:   F_GetFieldCont -- Read the content of a field

Synopsis:   F_GetFieldCont Field/A,Record/K/N
   rc = Success
   result = Content


Function:   Reads the content of the specified Field in the active or
   specified record and returns it in result.  May be only called in
   record mode.  This command may be also called in virtual state.

Inputs:   FieldId - ID of Field
   Record - Number of record (Fiasco 1.2)


Results:   rc = 0 - everything Ok, result will be the content
   rc = 5 - no record active
   rc = 10 - arg missing, or unknown ID.
    result - is equal to the current content of the field, if rc = 0.
   The format:
   String - the string itself.
```

```
   Integer - the number itself.
   Float - the fp number.
   Slider - the value of the slider.
   Cycle - the number of the active label.
   Date - the date in the format DD.MM.[YY]YY.
   Time - the time in the format HH:MM:SS.
   Extern - the string itself.
   Datatyp.- the string itself.


See also:
```

1.207 F_GetProjFullName

```
                F_GetProjFullName


Name:   F_GetProjFullName -- get the name of the current project

Synopsis:   F_GetProjFullName
   Result = Name

Function:   Returns the filename of the current project incl. path.

Note:   The path is relative to the current directory of Fiasco.

Inputs:

Results:   Name - Name of project incl. path.

See also:
                F_GetProjName
```

1.208 F_GetProjName

```
                F_GetProjName


Name:   F_GetProjName -- read the filename of the current project

Synopsis:   F_GetProjName
   Result = Filename

Function:   Returns the filename of the current project without path.
   This value may be used for
                F_SelectProj
                .

Inputs:   none
```

```
Results:   Result - Filename of the current project without path. A file
   must not necessarily exist. This is possible, if the name has been
   changed using Options and the project has not been saved.

See also:
               F_GetProjFullName
```

1.209 F_GetRecNum

```
               F_GetRecNum


Name:   F_GetRecNum -- Get the number of the current record.

Synopsis:   F_GetRecNum
   Result = Number_of_record

Function:   Returns the number of the active record in result. May be
   used to save the initial status of the project and to restore it at
   the end using
               F_GotoRec
               .

Inputs:   none

Results:   Result = Number of the record. Note that filters and other
   operations may change the record numbers.

See also:
```

1.210 F_GotoFirstRec

```
               F_GotoFirstRec


Name:   F_GotoFirstRec -- activate the first record

Synopsis:   F_GotoFirstRec

Function:   activates the first record. If the current project does not
   contain any records, nothing will happen.  Equivalent with
               Record/First
                  . May be only called in
               record mode
               .

Inputs:   none

Results:   none
```

```
See also:
```

1.211 F_GotoNextRec

```
               F_GotoNextRec


Name:   F_GotoNextRec -- activate the next record

Synopsis:   F_GotoNextRec

Function:   Activates the record after the active one. If the active
   record is the last record or the current project contains no records,
   nothing will happen.  Equivalent with
               Record/Next
               . May be only called
   in
               record mode
               .

Inputs:   none

Results:   none

See also:
```

1.212 F_GotoLastRec

```
               F_GotoLastRec


Name:   F_GotoLastRec -- activate the last record

Synopsis:   F_GotoLastRec

Function:   Activates the last record. If the current project does not
   contain any records, nothing will happen.  Equivalent with
               Record/Last
                  . May be only called in
               record mode
               .

Inputs:

Results:

See also:
```

1.213 F_GotoPrevRec

```
               F_GotoPrevRec


Name:   F_GotoPrevRec -- activate the previous record.

Synopsis:   F_GotoPrevRec

Function:   Activates the record, which precedes the active record. If
   the active record is the first record, nothing will happen.
   Equivalent with
               Record/Previous
               . May be only called in
               record mode
               .

Inputs:   none

Results:   none

See also:
```

1.214 F_GotoRec

```
F_GotoRec


Name:   F_GotoRec -- activate a record.

Synopsis:   F_GotoRec Record/A/N

Function:   Activate the record, whose number has been given as arg. If
   the number was invalid, do nothing.

Inputs:   RecordNumber - The number of the record. Please note, that
   sorting, adding or removing records or filters may change the record
   numbers.

Results:

See also:
```

1.215 F_GotoRecReq

```
                F_GotoRecReq


Name:   F_GotoRecReq -- open the
                Goto-Requester
                Synopsis:   F_GotoRecReq

Function:   Does exactly the same as
                Records/Goto
                . May be only called in

                record mode
                .

Inputs:   none

Results:   none

See also:
                F_GotoRec

                Record/Goto
```

1.216 F_Import

```
                F_Import


Name:   F_Import -- Import data

Synopsis:   F_Import File/A,RecStart/K,RecEnd/K,RecSep/K,FieldStart/K,
   FieldEnd/K,FieldSep/K,SkipLines/k,StartLine/N/K,FirstRecIDs/S,
   AppendFields/S
   rc = Success

Function:   Calls the import function of Fiasco. The specified file will
   be imported into the current project using the specified parameters.
   For more information on import and export see section

                Import and Export
                .    You may also use the escape sequences of Fiasco.
   If you do not specify a parameter, it will be empty.

Inputs:   File - Name of File
   RecStart,RecEnd,RecSep,FieldStart,FieldEnd,FieldSep - the structuring
   characters
   SkipLines - Comment introducer
   StartLine - Length of initial comment
   FirstRecIDs - First Record contains IDs
   AppendFields - Append new fields


Results:   rc = 0, if everything went well
```

```
Notes:   The option Overwrite old project of the import requester is not
   directly supported. You have to emulate it using
               F_ClearProject
               .

See also:
               F_Export
               , Chapter Import and Export
```

1.217 F_IsMarked

```
F_IsMarked


Name:   F_IsMarked -- Is the record marked?

Synopsis:   F_IsMarked Record/N
   rc = IsMarked

Function:   Looks, whether the current or the specified record is marked.
   If it is not marked, 5 is returned.  This command may be also called
   in virtual state.

Inputs:   Record - Number of record, if not specified, current record is
   used.

Results:   rc = 0: Record marked, = 5: Record not marked, > 5: other
   error

See also:
```

1.218 F_IsVirgin

```
               F_IsVirgin


Name:   F_IsVirgin -- Is the project unchanged?

Synopsis:   F_IsVirgin
   rc = Is_Virgin

Function:   Tests, whether the current project has been changed since the
   last saving. If it has been changed, Quit, Erase, Load and so on, will
   put an requester.

Inputs:   none

Results:   rc = 0 - Unchanged
```

```
   rc = 5 - Changed

See also:
                F_MakeVirgin
```

1.219 F_LoadDTObject

```
F_LoadDTObject


Name:   F_LoadDTObject -- Load the contents of a datatypes field

Synopsis:   F_LoadDTObject Field/A

Function:   Loads the contents of a datatypes field, which was
   "deferred".

Inputs:   Field - ID of datatypes field

Results:   The contents are loaded

See also:
```

1.220 F_Locate

```
F_Locate


Name:   F_Locate -- Locate the Cursor

Synopsis:   F_Locate X/A/N,Y/A/N

Function:   Sets the cursor at the given position. At this place the next
   mask operation will happen. May be only called in mask mode.

Inputs:   X - X-Coordinate
   Y - Y-Coordinate

Results:

Bugs:   Currently not particularly useful, because there are no direct
   commands for manipulating the mask.

See also:
```

1.221 F_LockGUI

F_LockGUI

Name: F_LockGUI -- Make the GUI not accessible by the user.

Synopsis: F_LockGUI

Function: Locks the GUI of Fiasco. The pointer will appear as a "wait clock". After locking the GUI, the ARexx script can run, without the danger of being influenced by the user. Before the script ends,

F_UnlockGUI
must be called in order to give the control back to the user. F_LockGUI and F_UnlockGUI may be nested.

Inputs: none

Results: none

Note: Make sure, that your scripts unlock the GUI in every case before exiting. Use signal commands to catch errors or breaks. For example:

```
/* test.rexx */

address FIASCO
options results

signal on syntax
signal on halt

F_LockGUI        /* Lock the GUI */

/* your code */

F_UnlockGUI      /* Unlock the GUI */

exit             /* And finish */

Syntax:
Halt:

F_UnlockGUI
exit
```

However, if a script leaves Fiasco locked, you may the following script, which is also available in the file ARexx/UnlockGUI.rexx:

```
/*
 * Fiasco will complain once,
 * if ARexx-Debug is activated
 */

address FIASCO
```

```
     do forever

         F_UnlockGUI

         if rc ~= 0 then break

     end



See also:
                F_UnlockGUI
```

1.222 F_MakeVirgin

```
                F_MakeVirgin


Name:   F_MakeVirgin -- Say Fiasco, that the current project is unchanged

Synopsis:   F_MakeVirgin

Function:   Pretends, that the current project is unchanged. This
   prevents certain procedures (Erase, Load, Quit,...) to put up a
   requester.

Inputs:   none

Results:   A project, that thinks, it has not been changed since the last
   saving.

Note:   The ARexx commands of Fiasco 1.1 provide direct arguments to
   suppress these warnings. Because of that, this function has no real
   meaning. It is recommended not to use this command, in order not to
   confuse the user.

See also:
                F_IsVirgin
```

1.223 F_MarkAllRecords

```
                F_MarkAllRecords


Name:   F_MarkAllRecords -- Mark all Records

Synopsis:   F_MarkAllRecords

Function:   Marks all records in the current project. They will be
```

```
   displayed highlighted in the list. Does exactly the same as

                Records/Mark All
                .

Inputs:

Results:

See also:
                F_UnmarkAllRecords

                F_MarkRecord
```

1.224 F_MarkMatch

```
                F_MarkMatch


Name:   F_MarkMatch -- Mark records, which match with a pattern

Synopsis:   F_MarkMatch Field/A, Blur/K, Pattern/F/A

Function:   Marks all records, which match with the given pattern in the
   given field. Operates similar to the filter. F_MarkMatch clears the
   marks of the records, which don't match.

Inputs:   Field -- The ID of the field, which will be examined
   Blur -- Takes the blurfactor of the comparison. Specify only, if you
   want to do blurred search.
   Pattern -- The pattern to search for.

Results:

See also:
                F_MarkRecord

                F_ToggleAllMarks
```

1.225 F_MarkRecord

```
                F_MarkRecord


Name:   F_MarkRecord -- Mark a record

Synopsis:   F_MarkRecord Record/N

Function:   Marks a record in the current project. It will be displayed
   highlighted in the list.  This command may be also called in virtual
```

```
   state.

Inputs:   Record/N -- Optional, if given the record specified by it's
   number will be marked. Otherwise, the current record will be marked.

Results:

See also:
                F_UnmarkRecord

                F_MarkAllRecords
```

1.226 F_NewProject

```
                F_NewProject


Name:   F_NewProject -- Open a new project window

Synopsis:   F_NewProject

Function:   Opens a new project. A new window is opened and activated. It
   is then entirely empty. Does exactly the same as
                Project/New
                .

Inputs:   none

Results:   none

Bugs:   Should claim on error.

See also:
```

1.227 F_OpenList

```
                F_OpenList


Name:   F_OpenList -- Open the
                list window
                Synopsis:   F_OpenList

Function:   This command is equal to activating the menuitem
                Control/List
                   .

Inputs:
```

```
Results:

See also:   Control/List
                F_CloseList
```

1.228 F_OpenProject

```
                F_OpenProject


Name:   F_OpenProject -- Load a project

Synopsis:   F_OpenProject File/A
   rc = Success

Function:   Tries to read a fiasco project into the current project
   window. The data, which are currently in the window will be freed
   without any request.

Inputs:   Name - Filename of the project

Results:   rc = 0, if everything went Ok,
   = 10, if argument is missing or file cannot be loaded.

See also:
                F_OpenProjectReq
```

1.229 F_OpenProjectReq

```
                F_OpenProjectReq


Name:   F_OpenProjectReq -- Open the "Open Project" ASL requester

Synopsis:   F_OpenProjectReq

Function:   Does exactly the same as
                Project/Open
                . I'm too lazy to write
   this here again. :-)

Inputs:   none

Results:   none

Note:   The user may have canceled the request

See also:
                F_OpenProject
```

```
                Project/Open
```

1.230 F_OpenServiceWin

```
                F_OpenServiceWin


Name:   F_OpenServiceWin -- Open the service window

Synopsis:   F_OpenServiceWin

Function:   Opens the
                service window
                , if it is not already open.

Inputs:   none

Results:   none

See also:
                F_CloseServiceWin
```

1.231 F_OptionsReq

```
                F_OptionsReq


Name:   F_OptionsReq -- Open the
                options requester
                 for the current
   project

Synopsis:   F_OptionsReq

Function:   Does exactly the same as
                Project/Options
                Inputs:   none

Results:   none

See also:
```

1.232 F_Progress

```
                F_Progress


Name:   F_Progress -- give the user a sense of the duration of a
   operation

Synopsis:   F_Progress Done/A/N, Max/A/N

Function:   Displays a nice progress bar in the service window, as known
   of Sort or Open Project. You should reset the status gadget with

                F_ResetStatus
                 when the operation has completed.

Inputs:   Done -- the number of data items currently processed.
   Max -- the number of all data items.

Results:

See also:
                F_SetStatus
```

1.233 F_Quit

```
                F_Quit


Name:   F_Quit -- close the current project

Synopsis:   F_Quit Force/S

Function:   Closes the current project. If you do not specify Force, this
   command does exactly the same as
                Project/Quit
                . That means, it is
   possible, that a requester opens, which asks you whether you want to
   save the current project before proceeding or cancel.  To prevent
   this, specify the Force parameter. This will suppress all warnings. To
   find out, wheter the project is not saved, you may use
                F_IsVirgin
                .

Inputs:   Force -- suppress all warnings.

Results:   none

Notes:   If the current project is closed, another project will be
   activated, or, if there is no other project, Fiasco will be shut down.
   An ARexx script should not rely on the order, in which the next
   project will be activated.

See also:
```

1.234 F_RemAllRecords

```
F_RemAllRecords


Name:   F_RemAllRecords -- Delete all records of project

Synopsis:   F_RemAllRecords Force/S

Function:   Removes all records of the current project. If you do not
   specify the Force parameter, this command does exactly the same as
   Record/Remove all. That means, that a requester may show up, which
   will ask you, whether you really want to remove all records. To
   prevent this behavior, specify Force.  This command may only be called
   in record mode.

Inputs:   Force - suppress all warnings

Results:   A project without any records.

See also:
```

1.235 F_RemRecord

```
               F_RemRecord


Name:   F_RemRecord -- Delete the active record

Synopsis:   F_RemRecord Force/S

Function:   Removes the active record and activates the next. If you do
   not specify the Force parameter, this command does exactly the same as
   Record/Remove. That means, that a requester may show up, which will
   ask you, whether you really want to remove this record. To prevent
   this behavior, specify Force.  This function may only be called in
   record mode.

Inputs:   Force -- suppress all warnings.

Results:   none

See also:
               F_AddRecord
               ,
               Record/Remove Record
```

1.236 F_RequestChoice

```
F_RequestChoice


Name:   F_RequestChoice -- request a choice

Synopsis:   F_RequestChoice Body/A,Gadgets/A,Title/K
   result = Selection

Function:   Creates an intuition easy-requester with the specified
   parameters. Works very similar to the CLI command Requestchoice. The
   differences: Slightly different parameters, puts the requester up on
   Fiasco's screen.  This command may be also called in virtual state.

Inputs:   Body - Main text of requester.
   Gadgets - Gadgets at the bottom of requester. Each choice must be
   separated by a |.
   Title - Title of requester.

Results:   result - Number of selected gadget, 0 for the rightmost one.

See also:
```

1.237 F_RequestField

```
F_RequestField


Name:  F_RequestField -- request a field ID (1.2)

Synopsis:  F_RequestField Text/A/F
   rc = Success
   result = SelectedField

Function:  Opens a requester with a list of all fields of the active
   project. The requester can display an additional message given in the
   Text argument. The user can select one field and click on Ok or can
   Cancel the requester. This command may be also called in virtual
   state.

Inputs:  Text - Text to display in the requester. May contain newlines
   (*N).

Results:  rc = 0 if user clicked on Ok, = 5 if cancelled.
   result = ID of selected field if rc = 0.

See also:
```

1.238 F_RequestFile

```
F_RequestFile


Name:   F_RequestFile -- request a file

Synopsis:   F_RequestFile File,Pattern/K,Title/K,Savemode/S,Drawersonly/S,
   Noicons/S
   rc = Success
   result = SelectedFile

Function:   Puts up an ASL file requester. Works very similar to the CLI
   command Requestfile. The differences: Slightly different parameters,
   puts the requester up on Fiasco's screen.  This command may be also
   called in virtual state.

Inputs:   File - Initial File including path for the requester
   Pattern - Initial Pattern
   Title - Title for the requester
   Savemode - Activates savemode: Black background, no selection via
   doubleclick
   Drawersonly - Displays only Drawers
   Noicons - Filters Icons


Results:   rc = 0, if user selected a file, otherwise user canceled.
   result = selected file, if rc = 0

See also:
```

1.239 F_RequestNumber

```
                F_RequestNumber


Name:  F_RequestNumber -- Request a number (Fiasco 1.2)

Synopsis:  F_RequestNumber DefaultValue/N,Title/K,Text/K/A
   rc = Success
   result Requested_Number

Function:  Asks the user to input an integer number. He may cancel the
   request. You can supply additional information using the Text
   argument.

Inputs:  DefaultValue - Value of integer gadget on startup. Will be zero
   if not specified.
   Title - Optional. Window title of requester.
   Text - Additional text to display in requester. May contain newlines
   (*n). Please note that this argument is required and must be specified
   with a leading keyword. This is for compability with future versions
   of Fiasco which may not require the Text argument.
```

```
Results:  rc = 0 if user clicked on Ok otherwise not equal zero.
   result = final value of integer gadget if rc = 0.

See also:
                F_RequestString
```

1.240 F_RequestString

```
                F_RequestString


Name:  F_RequestString -- Request a string (Fiasco 1.2)

Synopsis:  F_RequestString DefaultValue,Title/K,Text/K/A
   rc = Success
   result Requested_String

Function:  Asks the user to input a string. He may cancel the request.
   You can supply additional information using the Text argument.

Inputs:  DefaultValue - Value of string gadget on startup. Will be empty
   if not specified.
   Title - Optional. Window title of requester.
   Text - Additional text to display in requester. May contain newlines
   (*n). Please note that this argument is required and must be specified
   with a leading keyword. This is for compability with future versions
   of Fiasco which may not require the Text argument.

Results:  rc = 0 if user clicked on Ok otherwise not equal zero.
   result = final value of string gadget if rc = 0.

See also:
                F_RequestNumber
```

1.241 F_ResetStatus

```
                F_ResetStatus


Name:   F_ResetStatus -- restores the normal informations in the status
   gadget

Synopsis:   F_ResetStatus

Function:   Sets the status gadget of the service window to the normal
   contents. This is RecNum / AllRecs in the Record Mode or X / Y in the
   mask mode. You should use this call to reset the status informations
   set with
                F_SetStatus
```

```
                .

Inputs:

Results:

See also:
```

1.242 F_SaveProject

```
                F_SaveProject


Name:   F_SaveProject -- Save the current project

Synopsis:   F_SaveProject

Function:   Save the current project under the old name on disk. Does
   exactly the same as
                Project/Save
                .

Inputs:   none

Results:   none

Bugs:   Does not inform the script about errors.

See also:
                F_SaveProjectReq
```

1.243 F_SaveProjectReq

```
                F_SaveProjectReq


Name:   F_SaveProjectReq -- Open filereq and save project under new name

Synopsis:   F_SaveProjectReq

Function:   Does exactly the same as
                Project/Save As...
                .

Inputs:   none

Results:   none

Note:   The user may have canceled the request
```

```
See also:
                F_SaveProject
```

1.244 F_SaveSettings

```
                F_SaveSettings


Name:   F_SaveSettings -- Save the current program settings.

Synopsis:   F_SaveSettings

Function:   Does exactly the same as
                Settings/Save Settings
                .

Inputs:   none

Results:   none

See also:   Settings/Save Settings
```

1.245 F_SelectProj

```
                F_SelectProj


Name:   F_SelectProj -- activate an already load project

Synopsis:   F_SelectProj Name/A
   rc = Success

Function:   Activates a project, which stays already in memory. The
   filename without path is used to identify the project. This may be the
   name, which has been obtained using
                F_GetProjName
                . All following
   commands refer to the new project.

Inputs:   Name - Name of project without path.

Results:   rc = 5, if project is already active.
   = 10, if argument is missing, or there is no such project.

See also:
                F_GetProjName
```

1.246 F_SetFieldCont

```
                F_SetFieldCont


Name:   F_SetFieldCont -- Change the content of a field.

Synopsis:   F_SetFieldCont Field/A,Record/K/N,Cont/A/F
   rc = Success

Function:   Sets the content of the specified field in the active or
   specified record to the specified content.  May be only called in

                record mode
                . This command may be also called in virtual state.

Inputs:   Field - Identificationname of the field
   Record - Number of record (Fiasco 1.2)
   Cont - New content of the Field. This arg takes the whole input
   inclusive spaces. The Interpretation of this arg depends on the
   fieldtype:
   String - is copied directly
   Integer - Numbers are read directly, other things are 0
   Float - dto.
   Boolean - 1 or TRUE = selected, 0 or FALSE = not selected
   Slider - Number is read. Bad numbers will be adjusted.
   Cycle - Number or name of label is taken.
   Date - Date in Format DD.MM.[YY]YY is taken.
   Time - Time in Format HH:MM:SS is taken.
   Extern - is copied directly
   Datat. - is copied directly

Results:   rc = 0 - no error
   rc = 5 - no record is active
   rc = 10 - missing arg or bad FieldId

See also:
```

1.247 F_SetMode

```
                F_SetMode


Name:   F_SetMode -- Select the editing mode

Synopsis:   F_SetMode Mask/S, Records/S
   rc = Success

Function:   Activates the specified mode for the current project.

Inputs:   Mask - activates
                mask mode
```

```
                .
   Records - activates
                record mode
                .
   Mask and Records are mutually exclusive.

Results:   rc = 0 - no error
   = 5 - the project was already in the specified mode
   = 10 - missing or bad arg

See also:
```

1.248 F_SetSearchField

```
                F_SetSearchField


Name:   F_SetSearchField -- Set the field to search

Synopsis:   F_SetSearchField Field/A
   rc = Success

Function:   Sets the field, which will be searched by
                F_FindFirst
                 and

                F_FindNext
                .

Inputs:   Field - The Id of the field, which shall be searched.

Results:   returns 10 in rc, if the argument is missing or the id is
   unknown. otherwise 0 is returned.

Notes:   You don't need this function for simple searching, because
   Fiasco 1.1 allows passing these parameters directly with the search
   functions.

See also:
                F_SetSearchPat
```

1.249 F_SetSearchPat

```
                F_SetSearchPat


Name:   F_SetSearchPat -- Set the pattern to search for

Synopsis:   F_SetSearchPat Pattern/A/F
```

```
   rc = Success

Function:   Sets the searchpattern for the active project. If you also
   have specified a search field using
               F_SetSearchField
               , you may search
   for matching entries using
               F_FindFirst
                and
               F_FindNext
               . The value
   will be also used in the search requester.

Inputs:   searchpattern - a string to search for

Results:   rc = 0 - everything ok.
   rc = 10 - no argument
   rc = 20 - systemfailure (no memory, etc.)


Notes:   You don't need this function for simple searching, because
   Fiasco 1.1 allows passing these parameters directly with the search
   functions.

See also:
```

1.250 F_SetStatus

```
               F_SetStatus


Name:   F_SetStatus -- display a status string to the user

Synopsis:   F_SetStatus String/A

Function:   Displays the given string in the status gadget of the

               service window
               . If the service window is not open, nothing will be
   done.

Inputs:   String -- the string to be displayed

Results:

See also:
               F_ResetStatus
```

1.251 F_Sort

```
                F_Sort


Name:   F_Sort -- Sort the records of the current project

Synopsis:   F_Sort Field/A/M,Descending/S

Function:   Sorts the records of the current project according to the
   alphabetical or equivalent priority of the contents of the specified
   fields.

Inputs:   Field -- The ID of a field to sort after. Starting with Fiasco
   1.2 you may specify several fields to sort after. The first field has
   the highest priority while sorting.
   Descending -- Specify this, if you want the sorting to be backwards.

Results:

See also:
                F_SortReq
                ,
                Compare/Sort
```

1.252 F_SortReq

```
                F_SortReq


Name:   F_SortReq -- Open the sort requester

Synopsis:   F_SortReq

Function:   Does exactly the same as
                Compare/Sort
                . May be only called in
   record mode.

Inputs:

Results:

See also:   Compare/Sort...
```

1.253 F_ToggleAllMarks

```
                F_ToggleAllMarks
```

```
Name:   F_ToggleAllMarks -- toggle the marks of all records.

Synopsis:   F_ToggleAllMarks

Function:   Clears the marks on records, which were marked and sets the
   marks on previously unmarked records. Does exactly the same as

                Records/Toggle All Marks
                .

Inputs:

Results:

See also:
                F_MarkRecord
```

1.254 F_UnlockGUI

```
                F_UnlockGUI


Name:   F_UnlockGUI -- Unlock the GUI of Fiasco

Synopsis:   F_UnlockGUI
   rc = Success

Function:   Unlock the GUI, which has been previously locked using

                F_LockGUI
                . The user has again access to Fiasco. F_LockGUI and
   F_UnlockGUI may be nested.

Inputs:   none

Results:   rc not equal 0, if no lock was present.

See also:
                F_LockGUI
```

1.255 F_UnmarkAllRecords

```
                F_UnmarkAllRecords


Name:   F_UnmarkAllRecords -- Clear the mark on all records.

Synopsis:   F_UnmarkRecord

Function:   Clears all marks of the records in the current project. They
```

```
   will be rendered in a normal appearance in the list. Does exactly the
   same as
                Records/Unmark All
                .

Inputs:

Results:

See also:
                F_UnmarkRecord

                F_MarkAllRecords
```

1.256 F_UnmarkRecord

```
                F_UnmarkRecord


Name:   F_UnmarkRecord -- Clear the mark on a record

Synopsis:   F_UnmarkRecord Record/N

Function:   Clears the mark on a record, which has be set previously by

                F_MarkRecord
                 or using the GUI. The record will be rendered in a
   normal appearance in the list.  This command may be also called in
   virtual state.

Inputs:   Record/N -- Optional number of record to unmark. If not given,
   the current record will be unmarked.

Results:

See also:
                F_UnmarkAllRecords

                F_MarkRecord
```

1.257 F_UserCommand

```
F_UserCommand


Name:   F_UserCommand -- Calls a userdefined command.

Synopsis:   F_UserCommand Command/N/A

Function:   Calls a command, which has been defined in the "User" menu.
```

```
Inputs:   Command - Number of command, counted from zero. If this number
   does not exits, nothing will be done.

Results:   none

Note:   This command is only for implementing a icon bar or similar
   things. It should not be used in normal scripts, because the command
   may change freely.

See also:
```

1.258 F_VirtualMode

```
F_VirtualMode


Name:  F_VirtualMode -- Is this script called from virtual mode?

Synopsis:  F_VirtualMode rc = Virtual

Function:  Tests, whether the running script is called by Fiasco in
   virtual mode or in normal mode.

Inputs:

Results:  rc = 0: virtual status
   rc <> 0: normal status


See also:  Section virtual fields
```

1.259 Example Projects

```
                Example Projects

The directory databases of the Fiasco distribution contains several
Fiasco projects. Some of them may be also used for own purposes.


                Addresses
                 Addressbook

                DatatypesDemo
                 Demonstration of datatypes Fieldtype

                FamilyTree
                 stores data about you ancestors.
```

```
Videos
 Catalog of video tapes

PictureDatabase
 Manages your pictures

FAQs
 Manages textfiles
```

1.260 Addresses

```
Addresses

The Address project can be used as a simple addressbook. It contains
fields for Name, Address, Phone, etc. The project uses relations to
translate the abbreviations of country names (like "I" for Italy) to the
long names.

    The fields for Phone, Fax or Zipcode are string fields, because they
also have to take characters like "/" or must have a leading "0" (which
would be swallowed by a integer field).

    An additional Idea would be to use relations to search for the name
of the city using the zip code.
```

1.261 Datatypes Demo

```
                Datatypes Demo

This project is a easy Demonstration of the
                Datatypes fieldtype
                , which
requires the datatypes.library. For this reason, it is only available for
users of Amiga OS 3.0 or higher. The mask contains three fields, which
can be used to display all Data, which have the correct datatypes
installed.

    Two fields have scrollbars at the bottom and at the right side. You
can use these scrollers to move the contents of the field. The
stringgadget below the display contains the name of the file. The gadget
with the arrow down at the left side of the string gadgets can be used to
open a filerequester for editing the filename.

    One field has a button marked with an 'S'. This button opens a
filerequester, which allows you to select a file, in which the currently
displayed data are saved in. Fiasco writes the data in IFF format.

    The field at the upper right has the 'immediate play' attribute,
which plays the data --- if playable --- directly after loading the data.

    The browsing between records may get a bit slower, because the data
are stored in an external file and must be loaded first.
```

1.262 FamilyTree

FamilyTree

The family tree consists of the projects "persons.fdb" and "families.fdb". "persons.fdb" contains all persons, which are used in the family tree. You may also enter sex, date of birth, etc. here.

These data are used by "families.fdb" with relations, to get names of spouses, children, etc. Additionally, there are fields for marriage and divorce. Caused by the intensive use of relations, this project only contains 10 "real" fields, which are stored on disk. The other 12 fields are loaded from "persons".

1.263 Videos

Videos

The video database can be used to manage your homevideo collection. The database consists of two projects: "movies.fdb" and "tapes.fdb". "Movies" takes the informations for each movie (Genre, Director, etc.). The field "Tape" connects each film with one tape, which can be found in "tapes". Here is the play length of each tape defined. An ARexx script calculates the left free space on the tapes.

1.264 Picture Database

Picture Database

This databases uses the Datatypes fieldtype and is only usable, if you have Amiga OS 3.0 or better. The Datatypes field has the "defer" attribute, which means, that Fiasco won't load the data immediately. To read the data, you have to click in the string gadget of the field and press Return.

The string field under the datatypes field takes a description of the picture.

The button "Scan directory" can be used to read a freely selectable directory in the database. "Show on screen" displays the picture on an own screen. This button currently only works, if you use absolute paths for the graphics.

1.265 FAQs Database

FAQs Database

This database manages textfiles. For displaying the files, it uses the program Most by Uwe Röhm. If you prefer another textdisplay program,

which has an ARexx port, you may adopt the ARexx scripts.

The string field at the top of the mask takes the name of the textfile. Under it there are three buttons to control the database. "Scan dir" reads a selectable directory into this database. "View" displays the currently active text file. The most complex button is "Search". It can be used to search through the all files in the database for a string. If you click on it, a window opens, which asks you for an string to search for. Then it asks for a record to start the search. If you simply hit enter, it will begin at the first record. Then you are asked, whether you want to search all records or only the marked ones. After that Fiasco asks you, whether you want to write the results to a file. Then the last option comes, "Interactive searching". If you activate this, you will be asked for every found string, if you want to display the file at this place. After that the searching starts. If you want to break the searching, simply hit Ctrl-C.

The database contains the data for the FAQ (Frequently Asked Question) files on the Meeting Pearls II CD-ROM. If you don't own the CD ROM, use Record/Delete all Records to get rid of the data.

1.266 All Searchpatterns

All Searchpatterns

Pattern	Supported field types	function
--no pattern--	all types	exact matching.
#?	String, Extern, Datatypes	An unknown string with undefined length.
?	String, Extern, Datatypes	An unknown character.
> x	Integer, Slider	A number, which is greater than x.
< x	Integer,	

```
                Slider
                            A number, which is less than x.

    >= x
                Integer
                ,
                Slider
                            A number, which is greater or equal x.

    <= x
                Integer
                ,
                Slider
                            A number, which is less or equal x.

    !=
                Integer
                ,
                Slider
                            A number, which is not equal x.




    Detailed descriptions are available with the
                field documentations
                .
```

1.267 Relation Checklist

```
Relation Checklist
```

- create key field "there". Optionally activate "unique key".
- create real field "there". In case of string, extern or datatypes, remember "max chars".
- save project.
- create key field "here". Must be the same type as "there".
- create real field "here". Must be the same type as "there". In the case of string, extern or datatypes, "max chars" must be equal.
- save project.
- open relation requester for real field "here".
- select key "here"
- select relation file

· select key and real field "there". If the correct field is not
displayed, check type and in case of string, extern or datatypes max
chars.

· select Ok

1.268 Implementation of the Clipboard support

Implementation of the Clipboard support

The menuitems
Cut Record
,
Copy Record
and
Paste Record
use the clipboard
to store data temporarily. The clipboard of the Amiga OS is meant to provide a interface for different programs to share certain types of data. To make this possible, the clipboard may only contain IFF data.

Fiasco uses unit 0 of the clipboard and stores its data in IFF-FTXT files with a specific format. Each field gets a separate chunk. In this chunk the field content is stored in ASCII format.

The order of the chunks depends on the internal field list of Fiasco. Fiasco also uses this order to find out, which data belongs to which field while pasting the clipboard-contents.

With most other programs, you cannot create such structured IFF-FTXT files. The pasting in other programs is better supported. For example the conclip- program pastes the data correctly, while MultiView displays only the first chunk.

1.269 Bugs

Bugs

If you find some bugs in Fiasco, send a detailed description to
me
.
Please include information about your processor, OS version and other configuration.

These bugs are currently known:

· The frame of the list window flashes sometimes in a weird way
under Kickstart 37.x

- ARexx seems to have problems with filenames, which contain spaces. The name is only interpreted to the the first space. This affects the full path, because Fiasco expands the name to the full path before calling ARexx scripts.

- Seems to leave sometimes some memory allocated.

- Produces with asl.library 40.6 and Kickstart 40.70 MungWall hits after closing a filerequester. I think this is a bug of asl or intuition but not of Fiasco.

1.270 To do

To do

Fiasco is of course not perfect, at all. Here is a list of all things, which will be perhaps added at a later point (no guarantee!). If you have an Idea, send it to
me
!

- Better scrolling in the mask window. I currently use GadTools gadgets, which have to be recreated if you want to change their positions. I plan to emulate the used gadgets.

- New searchfunction.

- Sorting should get faster.

- New ARexx commands: ReadRecord and WriteRecord. Should read all fieldcontents and put them in ARexx variables with the field id as names.

- New searchpatterns.

- Hiding of fields in the mask.

- Stringfields, which support multiple lines

- Reversed logic while searching and counting

- AppWindows for Datatypes and Extern Fields

- "Packing" of projects: search for unused fields and make used as small as possible.

- Checking, whether a similar record already exists (automatically)

- Ability to specify the order, how the fields are activated after a Return.

- Fiasco should not have to read the whole file. (To save memory)
- Iconify projects
- List fieldtype with "Add" and "Del"
- "ARexx hook" for extending relations
- Better support of mask mode in ARexx
- "Input only once" field attribute
- Ability to influence the order, string, integer, etc. fields are activated
- OpenNewProject function

1.271 How to get contact

How to get contact

Send gifts, ideas, bug reports, etc. to:

Nils Bandener
Dekanatsgasse 4
D-34369 Hofgeismar
Germany

Internet: Nils@dinoex.sub.org

1.272 Index

Index

#?
  Patterns

,
  Patterns

?
  Patterns

about menuitem
  Project/About...

add element menuitem
  Element/Add...

```
create backups menuitem
  Settings/Create Backups?

create icons menuitem
  Settings/Create Icons?

cursor
  F_Locate

cursor
  Fiasco's Graphic User Interface

cut record menuitem
  Record/Cut Record

cycle
  Cycle fieldtype

data structure
  Basic elements of a Database

datatypes
  Datatypes fieldtype

datatypes/animation
  Datatypes fieldtype

datatypes/immediate playing
  Datatypes fieldtype

datatypes/scrolling
  Datatypes fieldtype

datatypes/sound
  Datatypes fieldtype

datatypes/speeding up record changes
  Datatypes fieldtype

date
  Date fieldtype

debugging of ARexx scripts
  ARexx and Fiasco in general

delete all records menuitem
  Record/Delete all Records

Delete gadget
  Delete

delete record menuitem
  Record/Delete Record

descending
  Sort requester
```

```
display menuitem
  Settings/Display...

display options requester
  Display Options Requester

dragging
  Fiasco's Graphic User Interface

duplicate element menuitem
  Element/Duplicate

duplicate field menuitem
  Field/Duplicate Field

duplicate record menuitem
  Record/Duplicate Record

dynamic service win menuitem
  Settings/Dynamic ServiceWin?

edit body menuitem
  Control/Edit Body

edit element menuitem
  Element/Edit...

edit field menuitem
  Field/Edit Field...

edit filter menuitem
  Compare/Edit Filter...

edit foot menuitem
  Control/Edit Foot

edit head menuitem
  Control/Edit Head

edit relation menuitem
  Field/Edit Relation...

edit usermenu menuitem
  User/Edit...

edit usermenu requester
  Usermenu Requester

editing the print mask
  The Print Mask

editor menuitem
  Settings/Editor...

eepic
  Printing with ARexx
```

```
fonts
  Mask

foreign data
  Import and Export

formatstring
  Slider fieldtype

function keys
  Usermenu Requester

F_AboutReq
  F_AboutReq

F_ActivateField
  F_ActivateField

F_AddFieldReq
  F_AddFieldReq

F_AddRecord
  F_AddRecord

F_ClearProject
  F_ClearProject

F_CloseList
  F_CloseList

F_CloseServiceWin
  F_CloseServiceWin

F_ConvertField
  F_ConvertField

F_CountRecs
  F_CountRecs

F_CountReq
  F_CountReq

F_DupRec
  F_DupRec

F_Export
  F_Export

F_FilterReq
  F_FilterReq

F_FindFirst
  F_FindFirst

F_FindNext
  F_FindNext
```

```
F_FindPrev
  F_FindPrev

F_FindReq
  F_FindReq

F_GetFieldAttributes
  F_GetFieldAttributes

F_GetFieldCont
  F_GetFieldCont

F_GetProjFullName
  F_GetProjFullName

F_GetProjName
  F_GetProjName

F_GetRecNum
  F_GetRecNum

F_GotoFirstRec
  F_GotoFirstRec

F_GotoLastRec
  F_GotoLastRec

F_GotoNextRec
  F_GotoNextRec

F_GotoPrevRec
  F_GotoPrevRec

F_GotoRec
  F_GotoRec

F_GotoRecReq
  F_GotoRecReq

F_Import
  F_Import

F_IsMarked
  F_IsMarked

F_IsVirgin
  F_IsVirgin

F_LoadDTObject
  F_LoadDTObject

F_Locate
  F_Locate

F_LockGUI
  F_LockGUI
```

```
F_MakeVirgin
  F_MakeVirgin

F_MarkAllRecords
  F_MarkAllRecords

F_MarkMatch
  F_MarkMatch

F_MarkRecord
  F_MarkRecord

F_NewProject
  F_NewProject

F_OpenList
  F_OpenList

F_OpenProject
  F_OpenProject

F_OpenProjectReq
  F_OpenProjectReq

F_OpenServiceWin
  F_OpenServiceWin

F_OptionsReq
  F_OptionsReq

F_Progress
  F_Progress

F_Quit
  F_Quit

F_RemAllRecords
  F_RemAllRecords

F_RemRecord
  F_RemRecord

F_RequestChoice
  F_RequestChoice

F_RequestField
  F_RequestField

F_RequestFile
  F_RequestFile

F_RequestNumber
  F_RequestNumber

F_RequestString
  F_RequestString
```

```
F_ResetStatus
  F_ResetStatus

F_SaveProject
  F_SaveProject

F_SaveProjectReq
  F_SaveProjectReq

F_SaveSettings
  F_SaveSettings

F_SelectProj
  F_SelectProj

F_SetFieldCont
  F_SetFieldCont

F_SetMode
  F_SetMode

F_SetSearchField
  F_SetSearchField

F_SetSearchPat
  F_SetSearchPat

F_SetStatus
  F_SetStatus

F_Sort
  F_Sort

F_SortReq
  F_SortReq

F_ToggleAllMarks
  F_ToggleAllMarks

F_UnlockGUI
  F_UnlockGUI

F_UnmarkAllRecords
  F_UnmarkAllRecords

F_UnmarkRecord
  F_UnmarkRecord

F_UserCommand
  F_UserCommand

gadgets
  Mask

gadtools.library
  Mask
```

```
remove element menuitem
  Element/Remove

remove field menuitem
  Field/Remove Field

remove relation menuitem
  Field/Remove Relation

replace
  Replace

replace menuitem
  Compare/Replace...

replace requester
  Replace requester

RESULT
  ARexx and Fiasco in general

save as menuitem
  Project/Save As...

save as menuitem in print window
  Project/Save as...

save menuitem
  Project/Save

save menuitem in print window
  Project/Save

save settings as menuitem
  Settings/Save Settings as...

save settings menuitem
  Settings/Save Settings

saving disk space
  Relations

screenmode requester
  Requirements

screenmode requester
  Display Options Requester

search pattern
  Searching in a database

search requester
  Search requester

search requester/ARexx
  Searching with ARexx
```

```
statistic
  Project/Statistic...

statistic menuitem
  Project/Statistic...

stretching
  Stretching of the mask

string
  String Fieldtype

structure of Import/Export files
  Structure of Import/Export files

talking
  Settings/Talking?

talking menuitem
  Settings/Talking?

tape deck gadgets
  Mask Mode

TeX
  Printing with TeX

text
  Text fieldtype

there project
  Creating Relations

time
  Time fieldtype

toggle all marks menuitem
  Record/Toggle all Marks

tolerance
  Blurred Search

true
  Boolean Fieldtype

unmark all records menuitem
  Record/Unmark all Records

unmark record menuitem
  Record/Unmark Record

update relations menuitem
  Settings/Update Rels?

Use * as pattern menuitem
  Settings/Use * as Pattern?
```

```
use filter menuitem
  Compare/Use Filter?

usermenu requester
  Usermenu Requester

virtual fields
  Standard Attributes

wait clock
  F_LockGUI

write relations menuitem
  Settings/Write Relations?
```