FEC startup configuration via the AB/CO Controls Configuration Database.

E. Bracke (AB/RF-cs)

Table of Contents.

Table of Figures.

Introduction.

Here is given an overview of how to configure a Front End Computer (FEC) with application programmes and driver software which will be automatically started when the FEC is booted.

The overview is done by giving a series of screen captures in the order that was required for configuring the LEIR test machine `dleitst3` for the Fesa device classes that are responsible for the remote control of the LEIR accelerator by the PS operation crew. Of course, creating the configuration for LEIR is only used as an example; the same procedure is valid for any FEC configuration.

FEC startup configuration is done with a startup reference file, called `transfer.ref`, which is generated from data that was entered into an Oracle database. The database is called the 'Controls configuration database' and is managed by the AB/CO-DM section.

The generation of this startup reference file from the 'Controls configuration database' is described for the LEIR control frontend computers in a separate note (see: 'genCCode.pdf' and the other references in there) and can also be considered as valid for any FEC configuration.

We start this description with the sequence of actions that define a 'Program' macro for collecting startup parameters for the LEIR FEC application(s) one wants to incorporate in the computer's startup sequence. After having defined the 'Program' macro, its use for the definition of the overall startup sequence table in the 'Controls configuration database' for FEC `dleitst3` is shown.

This is followed with a sequence of screen captures that explain how to enable / disable loading at boot time of driver software for accessing the Bnldsp boards of a FEC. It is of interest when one wants to start the drivers manually (with a script) in e.g. their 'simulator' mode for tests of control (Fesa) software if no 'real' hardware is available or in case that a 'defined and known' hardware state is required.

This note is stored in the `BNLDSP/DOCUMENTATION` directory which can be found in the CERN CVS file repository, currently at:
http://isscvs.cern.ch/cgi-bin/viewcvs-all.cgi/?root=abrfcs
(and also available on Erik's home page at:
http://bracke.home.cern.ch/bracke/HTML/LEIR_Development/LEIR_Development.htm).

Note to the reader:

Several times reference is made in here to other notes which have been stored (like the current document) in the central CERN CVS repository, module: abrfcs/BNLDSP in the directory `DOCUMENTATION`. The reader is invited to extract these notes if more info were sought. However, I can not assume absolute responsibility for the truthfulness in these notes; not all are written by me. I did my best, they are offered for what they are. In my personal case: they are the result after painfull gathering of info for making a FEC autoboot my application programmes. The info comes from many AB/CO specialists **and** it made it work for LEIR for me…

TIA FYU
E. B.

Logging in to the FEC Configuration data base.

Logging on to the FEC Configuration database is done via the AB/CO Data Management section's home page at: `http://ab-div-co-dm.web.cern.ch/ab-div-co-dm/` and then selecting the hyper link: '`Controls Configuration Portal`'.
From the page that follows, under Data entry tools, select: '`AB Controls Configuration Applications`', which, after some time, brings up the Oracle 'Login Form' that you must fill out.

After submitting a first time, the form asks which application it should start. Select here: '`Hardware`'. Submitting once more starts the Oracle '`Hardware`' application.

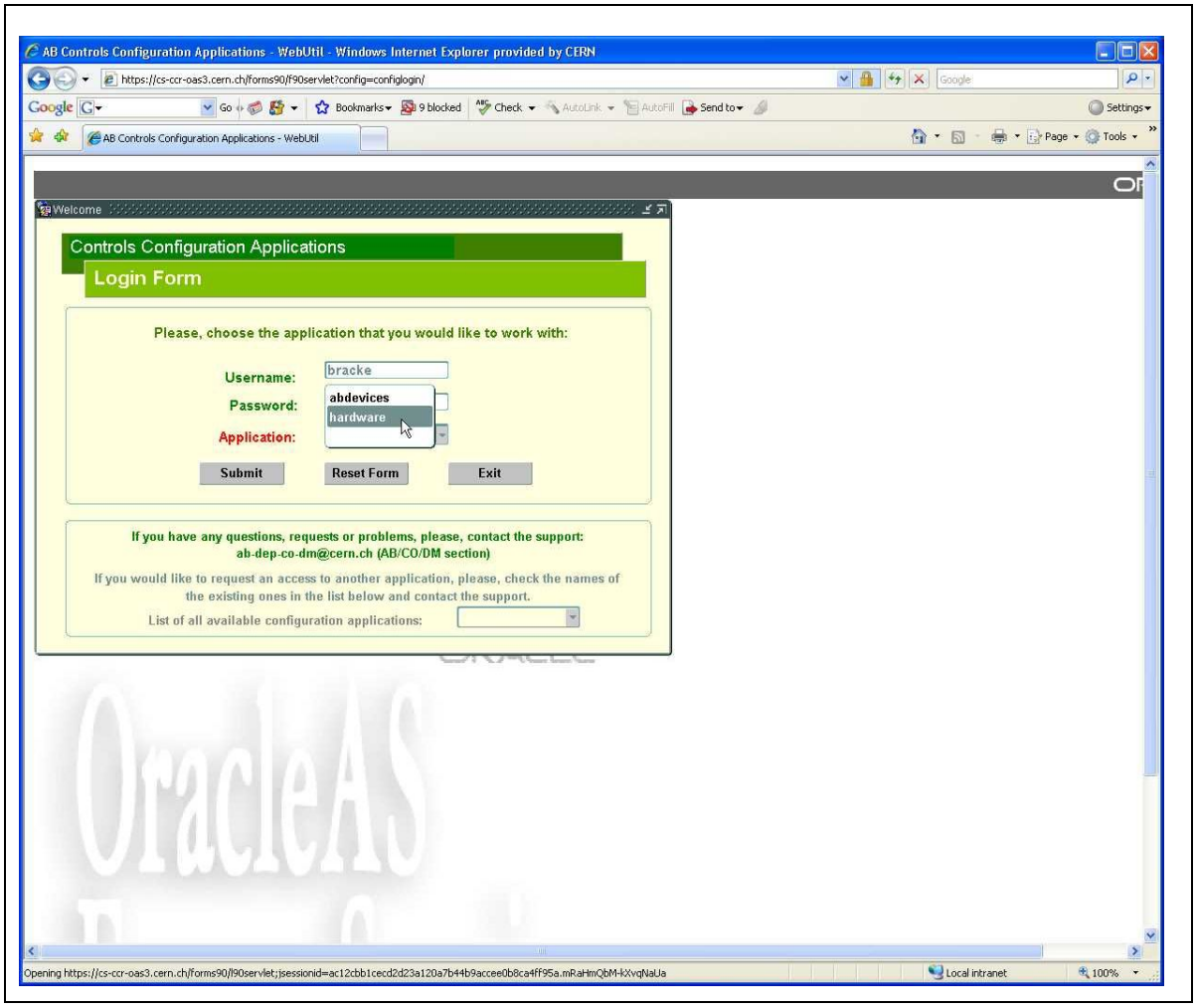

Figure 1 Login form with 'hardware' application select.

Creating a 'Program' (macro).

The FEC, when starting an application, needs to be instructed how the application wants to be launched (start parameters and work directory, priority etc.). By reading an entry line in its `transfer.ref` file, the computer has all the information for that application available.

Gathering, in a most generic (symbolic, parameterized) way, this information in a 'per application' fashion, is the task of the 'Program' macro. It is the 'Program' macro in the 'Configuration data base' which, when expanded, yields a line of text according to a `transfer.ref` specific format, in that file, for that application.

See the note: 'Starting User Programs on LynxOS, Linux and HP-UX computers.pdf' for details of this format.

The 'Configuration data base' caters for a record that takes these application parameters. Many application programmes for many FECs are started in the same way. Therefore we sometimes can use an already existing 'Program' macro, but sometimes we must create one specifically for our application. Here we describe the way how to create a new 'Program' macro.

We start in this case from the Oracle 'hardware' application by selecting from its menu the '`Definitions`' dropdown list and then '`# Dsc Program definitions`'.

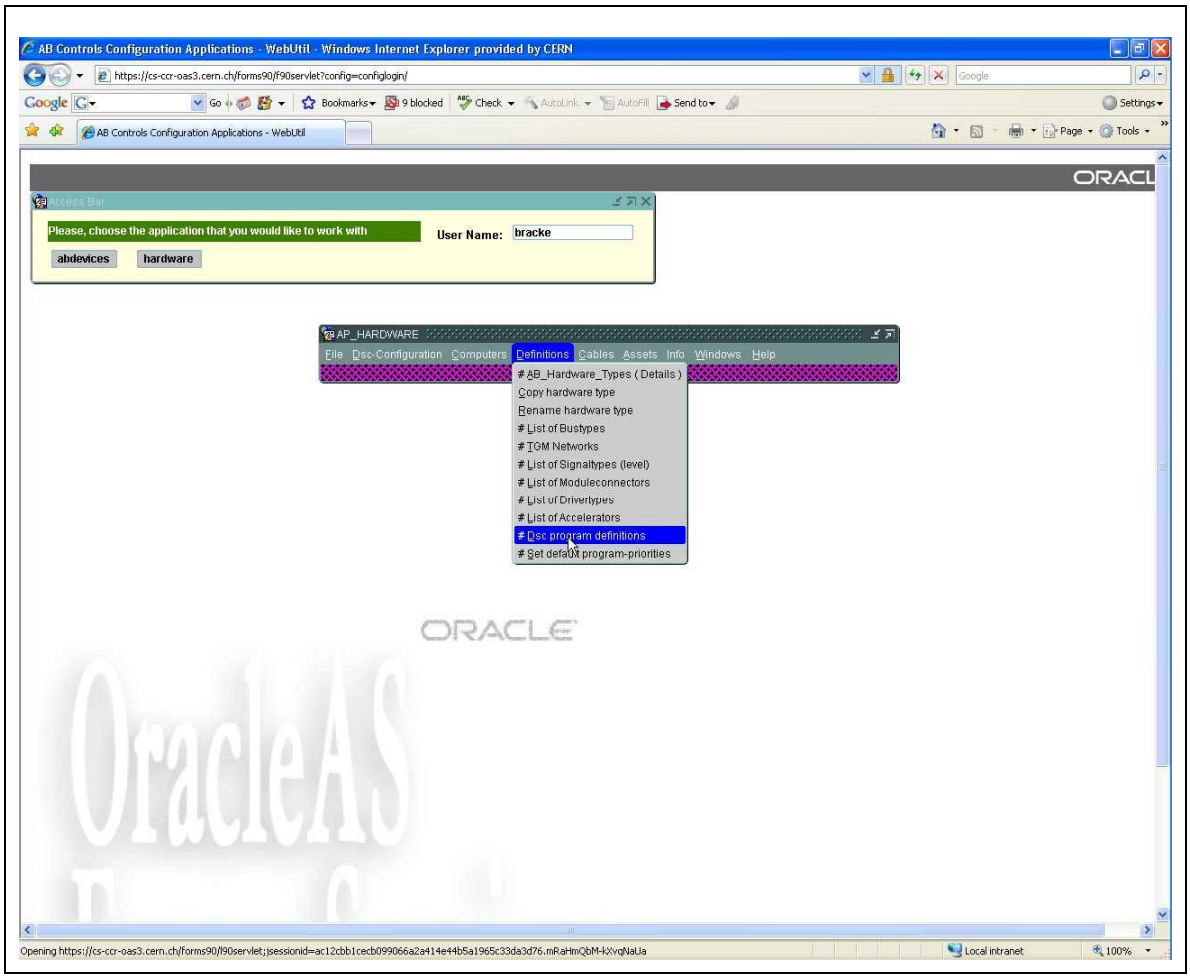

Figure 2 Selecting the 'Program definition' form.

A database query form is opened in which the user can formulate a database query for an already existing macro from which he could 'clone' the info while creating a new macro of his own. The 'List' menu offers another query window for finding an existing 'Program' to be entered in the first query form. Remember that in Oracle-ish the '%' character designates a wild card for 'any suite of characters'.

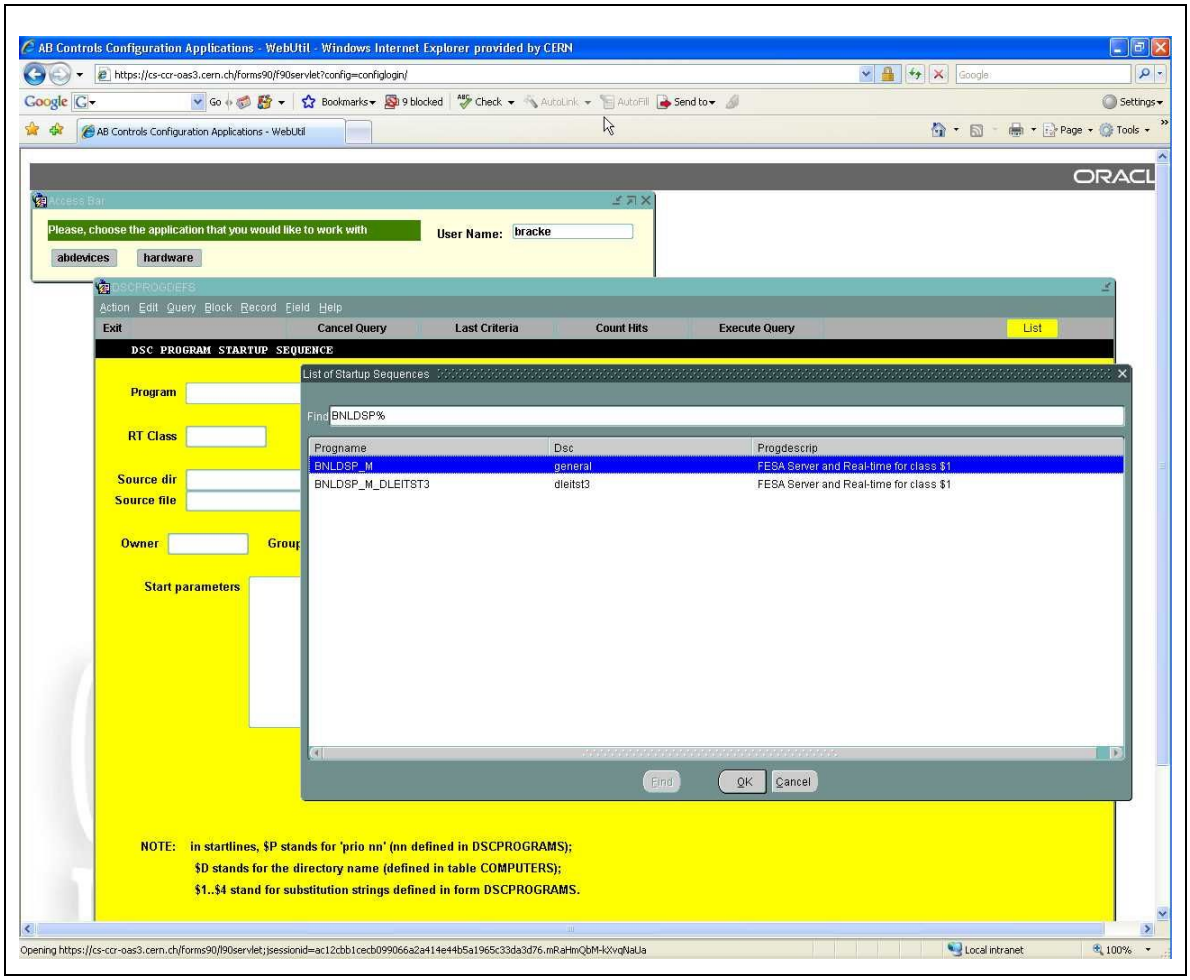

Figure 3 Database query for a 'Program' macro.

Once a 'Program' defined we can execute the database query and a new form will open with the selected macro and its current parameter definitions. This window gives various database editing facilities for changing and updating the 'Configuration data base' with the customized 'Program' macro record on screen.

For the interpretation of the various fields in this record in the `transfer.ref` file, see the note: 'Starting User Programs on LynxOS, Linux and HP-UX computers.pdf'.

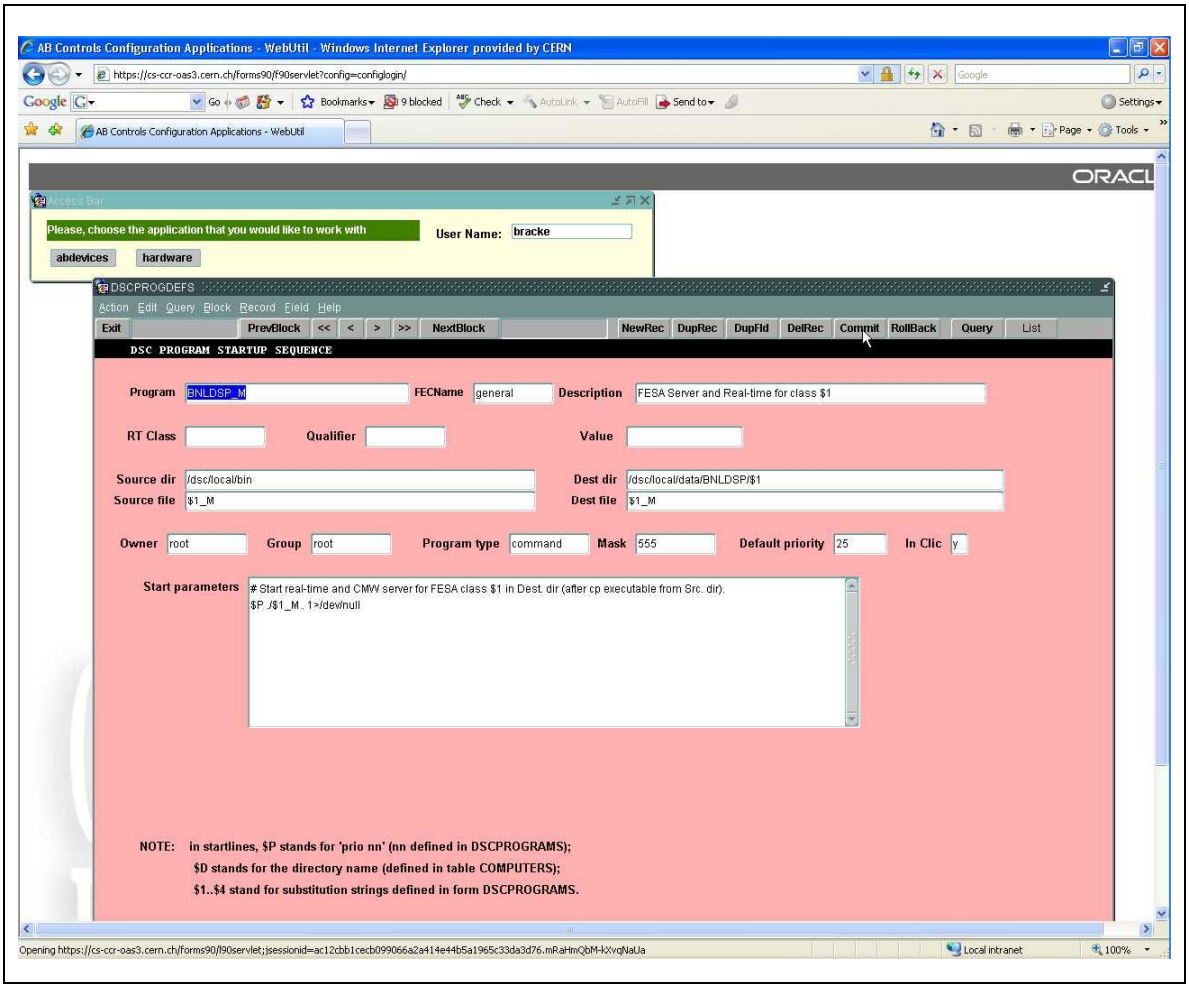

Figure 4 The 'Program' macro modification record.

An important detail of a Program macro is the definition of the 'In Clic' field. When defining it here as 'Y'-es, later while using the macro, it will signal the `transfer.ref` interpreter that the programme in question must be launched as a server rather than being launched via a shell. This has the effect that in the FEC's process table only <u>one</u> slot will be used (the one for the programme itself) in stead of 2 slots: one for the shell and one for the programme launched by it (FEC resource preservation). Indeed, most of the at startup launched programmes will stay 'on stack' indefinitely; never ending. It will, however, always be possible to not incorporate the programme in 'clic suveillance' by defining its startup sequence line's 'C'-lic column field with 'N'-o. (See also the info at: Figure 7 Definition of the startup sequence in the 'Configuration data base'.)

After 'Commit'-ting the created (by renaming the from the database retrieved template macro e.g.) or modified an existing 'Program' macro, it will be stored in the 'Configuration data base' and ready for use for the definition of a FEC's startup sequence.

Setting up the FEC application startup sequence.

When starting up, the FEC reads, one by one, the lines in its `transfer.ref` file. If a 'simple' user, like e.g. the AB/RF group, receives a FEC, the AB/CO group already has configured its infra structure (CPU, timing hardware etc.), often with the required driver software inclusive. This is reflected in the first few lines of the `transfer.ref` file. Some aspects (inhibiting driver load during startup e.g.) are discussed at the end of this note. Of interest in the current discussion is notably the part where **application** programmes are started, usually the last few lines available from the file.

Definition of the order in which events happen during startup of a FEC, is the task of the 'DSC Programs' table. It is this table in the 'Configuration data base' which, when extracted, yields the lines of text according to the `transfer.ref` specific format, in that file, for **all** initialization (drivers etc.) as well as for the applications in a FEC that must be auto-booted.

See once more the note: 'Starting User Programs on LynxOS, Linux and HP-UX computers.pdf' for details of this format.

Now we shall be making use of the before hand created 'Program' macro. Here we describe the way how to define a FEC's 'DSC Programs' table in the 'Configuration data base'.

We start in this case from the Oracle 'hardware' application by selecting from its menu the '`Dsc Configuration`' dropdown list and then '`# Dsc Startup`'.

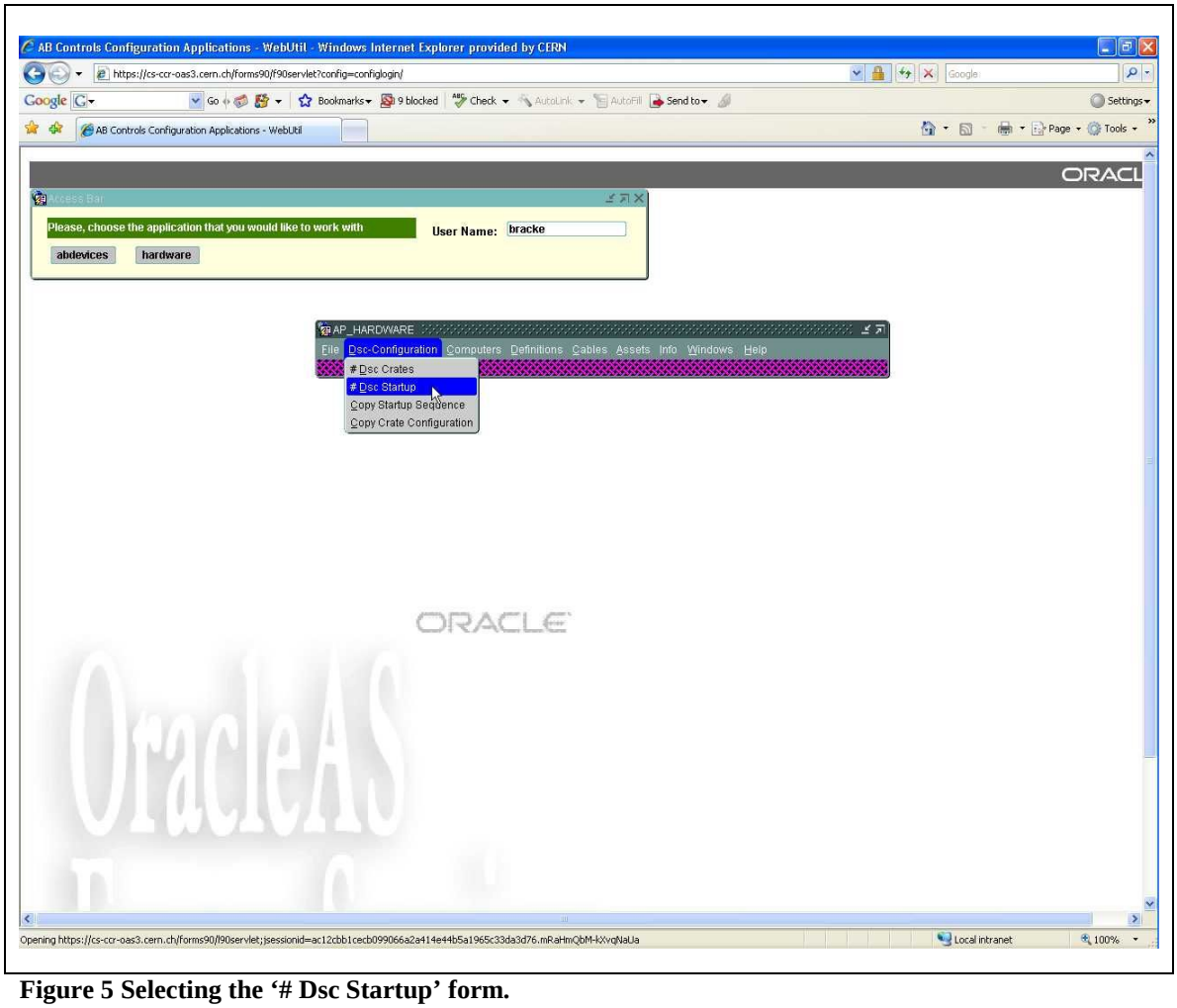

Figure 5 Selecting the '# Dsc Startup' form.

A new query form opens where the reader is invited to specify for which FEC (DSC called in the past, and even PCA before, nothing really changed, but that's another story…) he wants to define the startup sequence.

Note that we can only enter an already existing FEC from the ‘Configuration data base’. A new FEC must first be entered into the database by AB/CO first before its startup sequence can be configured.

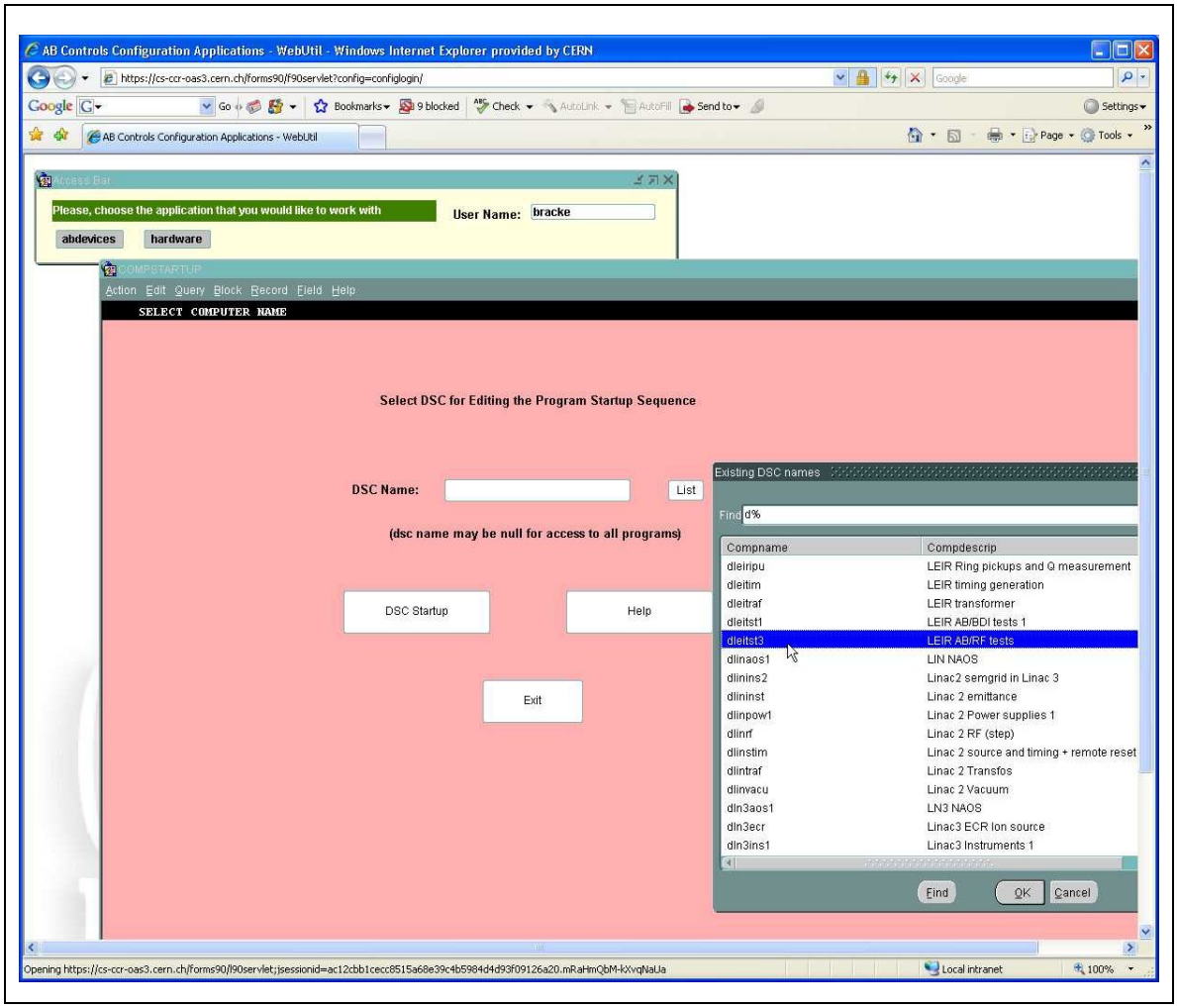

Figure 6 Selecting the FEC.

Select the required computer and click ‘OK’ to enter it in the ‘DSC name’ field. Button ‘DSC Startup’ then retrieves the startup table from the ‘Configuration data base’ in a new form.

We now can add / modify this 'Computer startup' form to our liking and eventually put it back into the database.

This is a short description of the columns:

- Dsc: The FEC for which it is concerned. Pre determined field, never changed.
- Seq: Ultimite order of the lines in the `transfer.ref` file
- Startup-name: Here we enter our 'Program' macro which we defined beforehand.
- C: 'Y' means that the programme will be suveilled by AB/CO's 'clic' tool.
- I: 'I' means that this line will be 'commented out' in `transfer.ref`; effectively inhibiting its startup execution.
- $1, 2, 3, 4: Freely useable substitution parameters for use in the 'Program' macro.

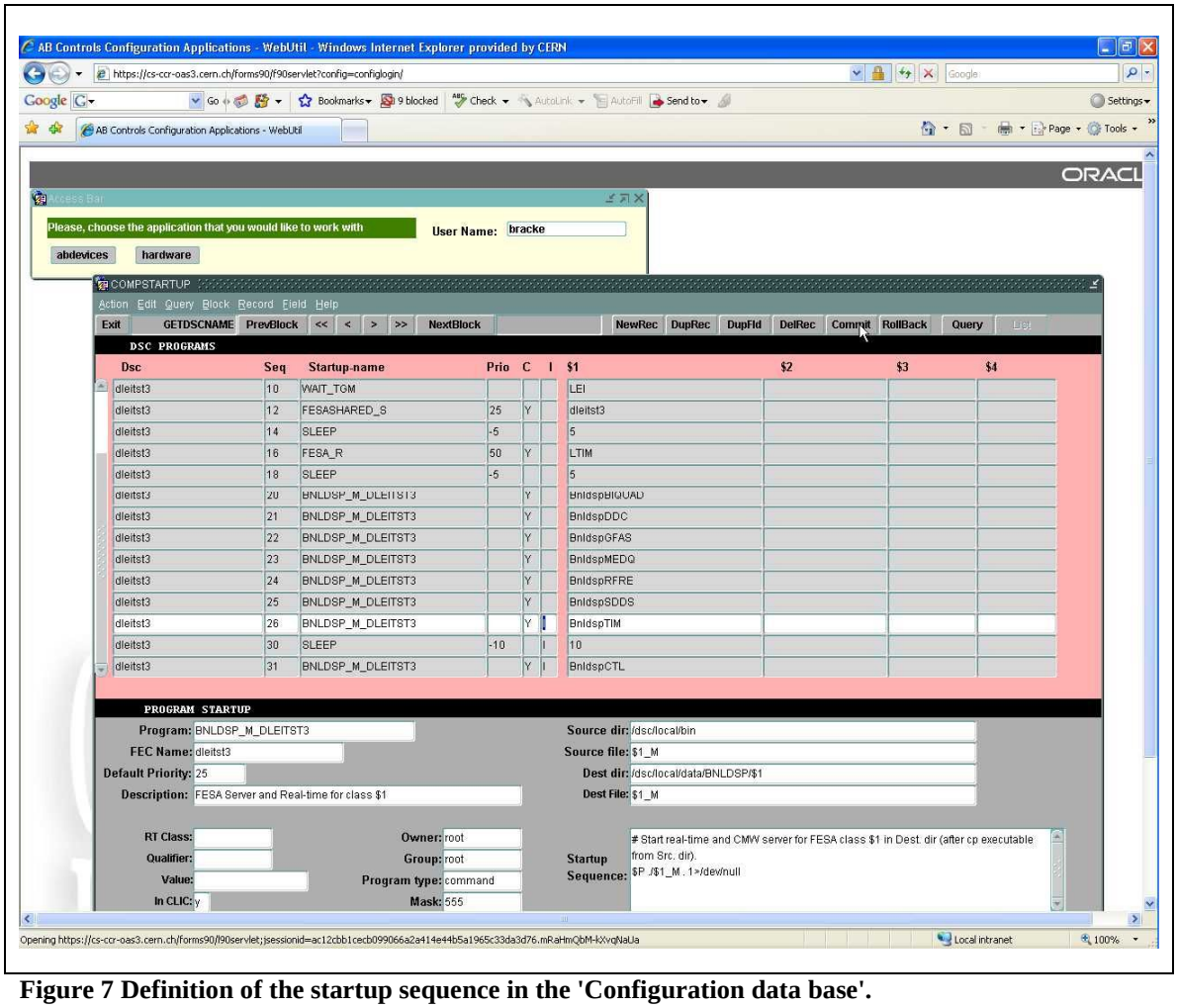

Figure 7 Definition of the startup sequence in the 'Configuration data base'.

Important note:
For 'Clic' surveillance it it mandatory that the Program macro used for this startup sequence entry has its 'In CLIC' field also set to 'Y'-es, otherwise no Clic surveillance will be performed during operation. (Thanks Frode and Alastair for these important details. See also the info at: Figure 4 The 'Program' macro modification record.)

Finally, click 'Commit' to upload the changes into the 'Configuration data base'

Creating a new `transfer.ref` file is explained, for the LEIR control system, in note: 'genCCode.pdf' and the ones that are referenced in there.

Driver enable / disable information.

Consider the FEC startup sequence screen capture below. Of interest is here the program macro `IOCONFIGINSTALL` which defines, amongst other things as well, the driver configuration for this FEC and will be translated in a few lines in the `transfer.ref` file when the latter is generated from the database.

Inspection and modification of the data, used by this macro, allows us to control which drivers will be included in the (system) ioconfig load file that will be read when the entries (several lines) in `transfer.ref` represented by `IOCONFIGINSTALL` are parsed during boot time of the FEC.

Generally, for us, 'simple users' of the Configuration Database, only the possibility of enabling / disabling of our specific drivers during boot time is the only interest. (E.g. if we want to install / uninstall manually a driver or, when we want to start a driver in 'simulation' mode.) We would do a thing like this if we were testing Fesa application software but we do not have 'real' hardware available. Indeed, all LEIR Fesa classes, but in particular BnldspTIM, BnldspCTL, BnldspMEDBUG, BnldspMEDQ and BnldspSCOPE, for which this feature is very useful, can be started with a commandline option that pre-loads a driver in simulation mode with 'dummy' acquisition data during class startup and that can subsequently be accessed as if 'real' hardware were accessed. Manually loading / unloading a driver can be accomplished with a set of batch scripts to be found in the central CERN CVS repository, module: abrfcs/BNLDSP, in the root directory.

This chapter shows the procedure how to do the manipulation of enabling / disabling the autoboot driver load for the LEIR Bnldsp board driver.

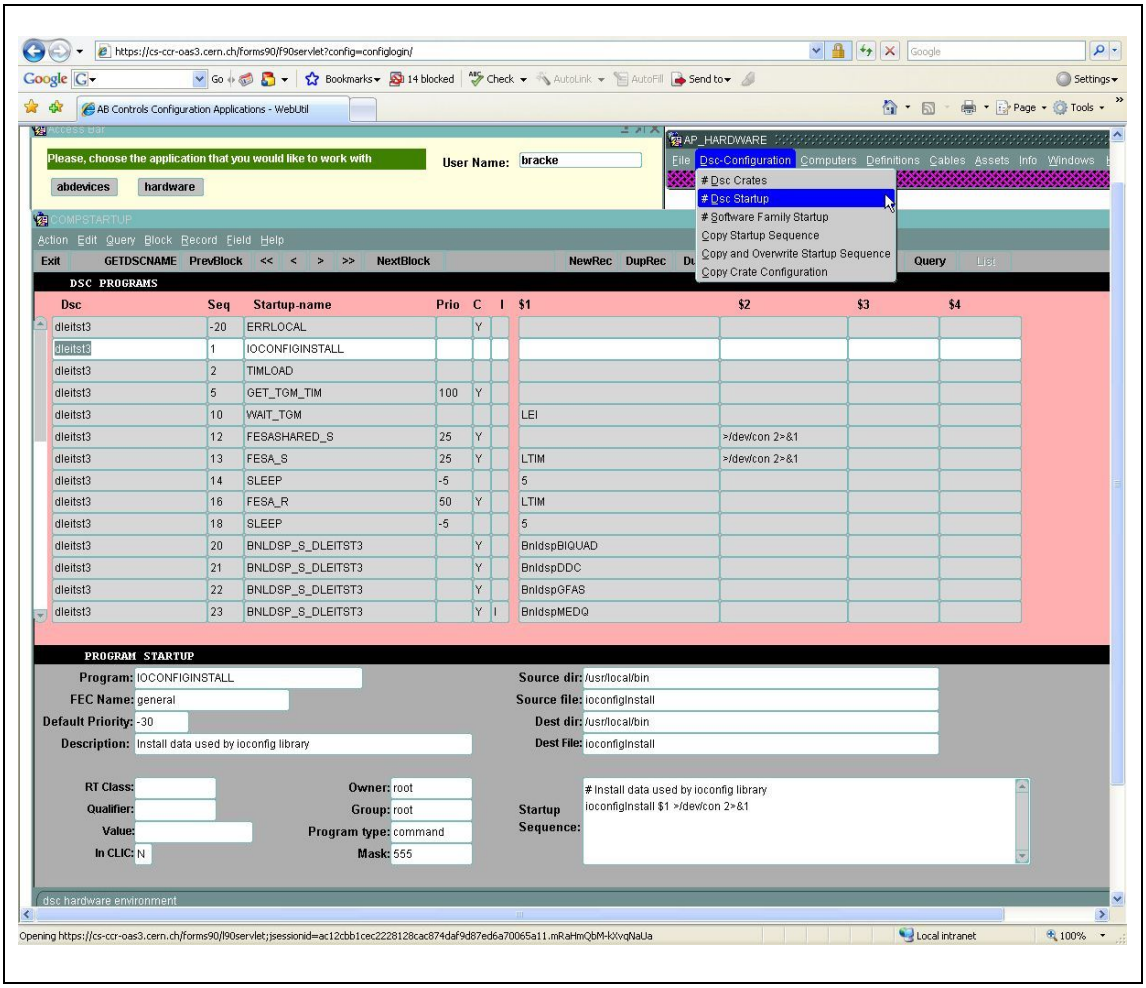

Figure 8 FEC startup IOCONFIGINSTALL macro.

First login onto the FEC Configuration database as explained in Figure 1 Login form with 'hardware' application select.; and start the Oracle 'hardware' application and we select from its menu's '`DSC Configuration`' dropdown list the item '`# Dsc Crates`'.

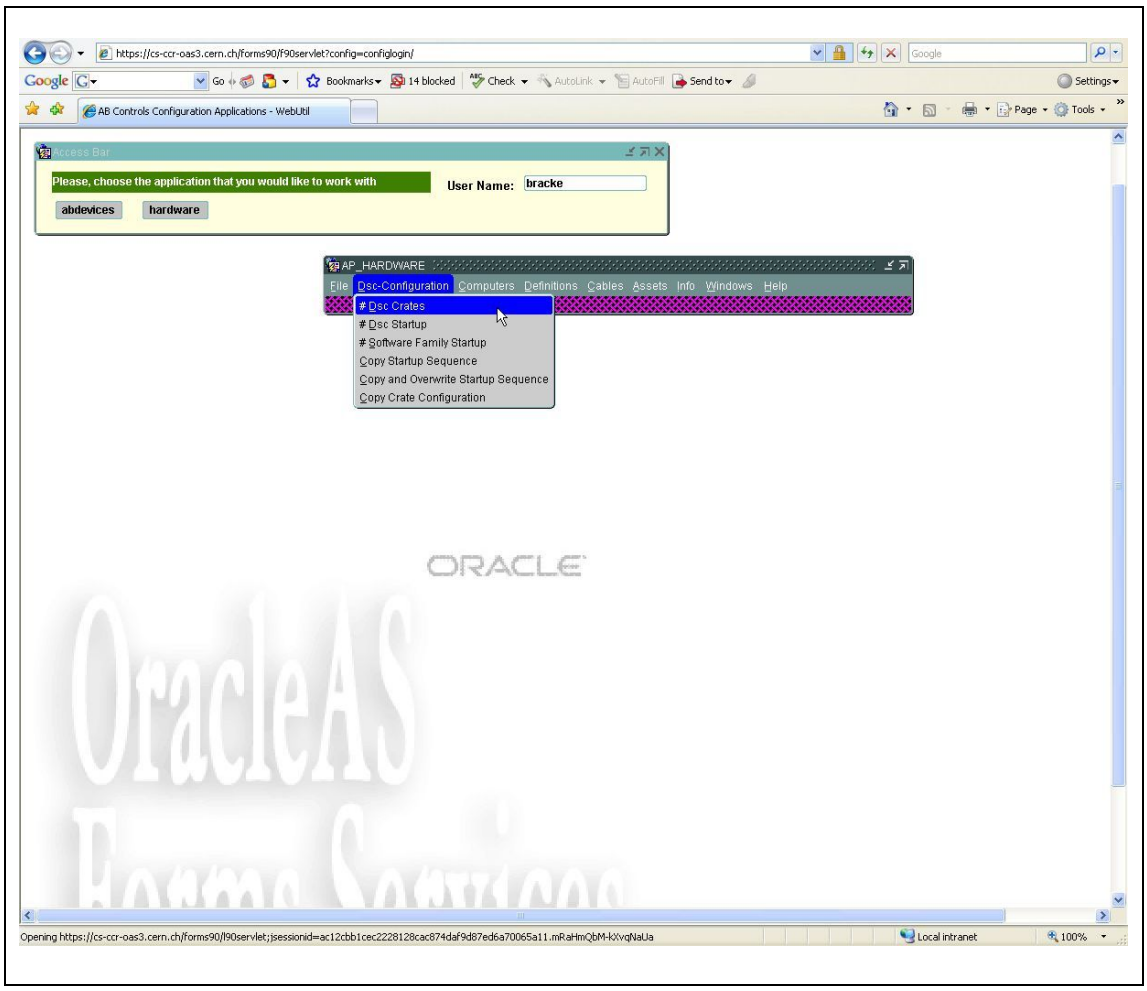

Figure 9 Selecting the '# Dsc Crates' form.

Then is asked the FEC from which information is required. We select e.g.:

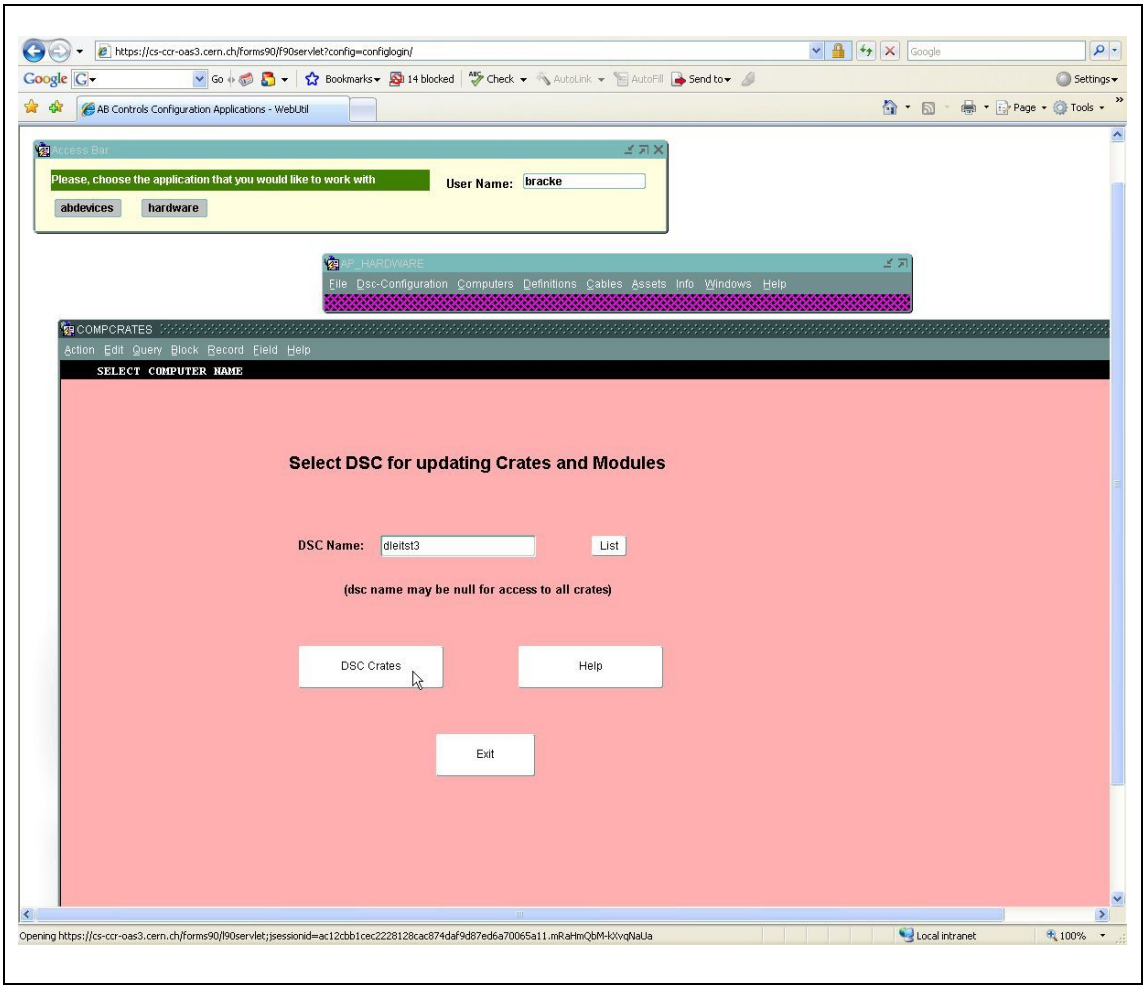

Figure 10 Selecting the FEC.

This pops up the screen below, where we have to ask for the ‘NextBlock’ to get all information of the selected FEC we need to see / modify.

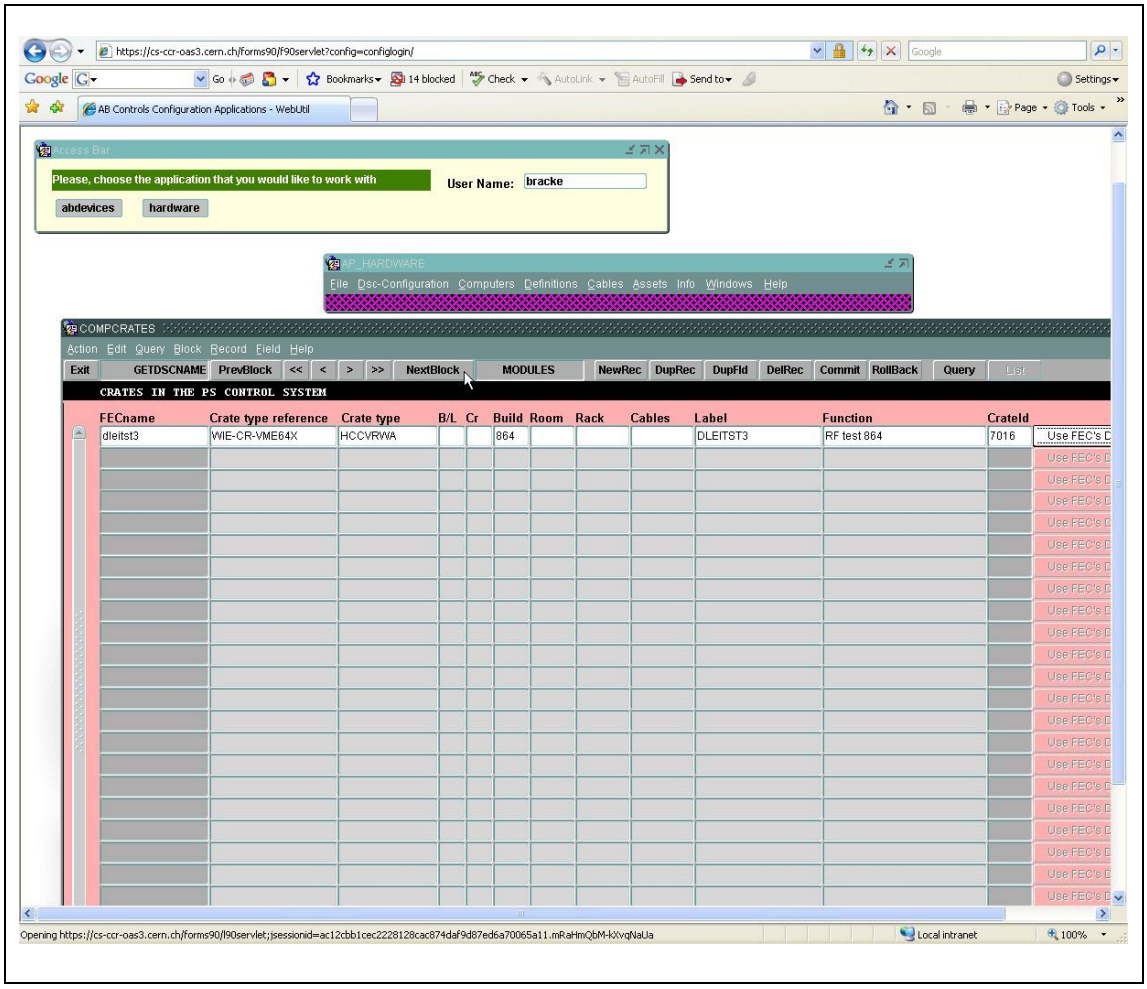

Figure 11 Crate 'modules' configuration (initial).

With the overview after ‘NextBlock’ we clearly see entries for 3 instances of the BNLDSP driver for the Bnldsp boards of the LEIR beamcontrol system.

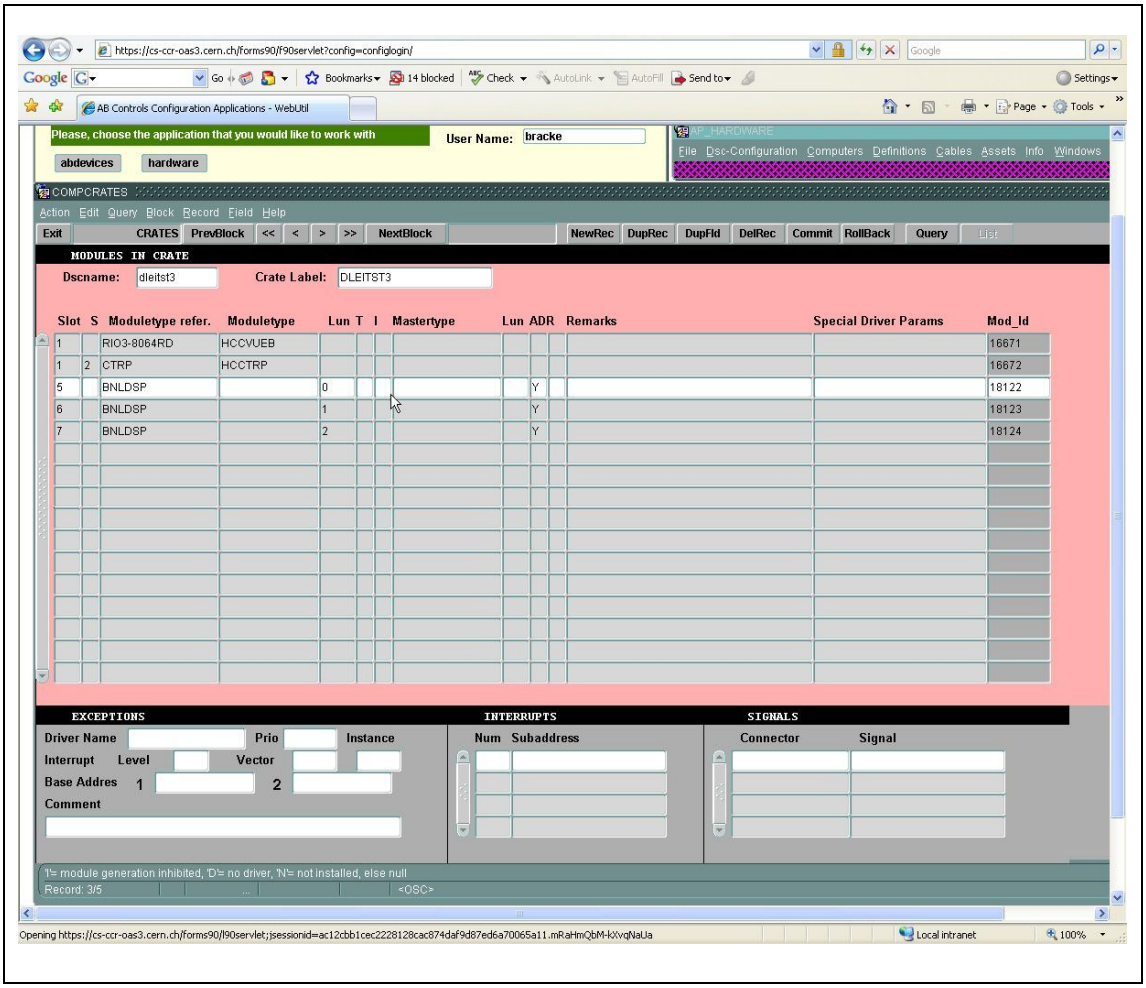

Figure 12 Crate 'modules' configuration (after 'NextBlock').

It is in the collumn, indicated by the mouse cursor, where one can put (or omit, like here) an ‘I’ flag if it is required to inhibit loading of instance (Lun) ‘0’ of the BNLDSP driver at FEC startup.

When changing, one should also do ‘Commit’ in order to update the database. Afterwards a new `transfer.ref` file must be created and installed. The procedure for this is explaned in document: ‘The generation of Bnldsp C object code modules and libraries for the LEIR Fesa BnldspXXX device classes.’ It can be found in the `DOCUMENTATION` directory of the Bnldsp C software at the CERN CVS repository.

Here is an excerpt of the `transfer.ref` file for dleitst3 that was created with all 3 instances of the driver inhibited:

```
#*****************************************************************************
# $Id: transfer.ref,v 1.38 2007/09/12 10:09:49 bracke Exp $
#*****************************************************************************
# WARNING: File generated from database. Can be overwritten at any time !
#

# ***** IOCONFIG Information *****

#   ln mln bus mtno module-type      lu W AM DPsz basaddr1   range1 W AM DPsz basaddr2   range2 testoff  sz sl ss
#+#  1   0 PCI  502 CTRP             0 N -- DP16       0        0 N -- ----       0             0  0  1  2
#+#  2   0 VME  186 RIO3-8064RD       0 N -- DP16       0        0 N -- ----       0       0       0  0  1 -1

#   ln sln bus mtno module-type      lu evno  subaddr    A1 F1     D1    A2 F2     D2
```

```
#
# ***** Program Startup before drivers *****
#
#
# start local errors output
#% prio 18   errlocal  >/dev/con 2>&1 &
#
#% cd /usr/local/drivers/sacvme; sacvmeinstall -R0 -M0 -V254 -L2
#% cd /usr/local/drivers/ctr; ctrinstall
#
#% upfiles -all
#% wreboot -all
#% end
#
# ***** Program Startup after drivers *****
#
#
# Install data used by ioconfig library
/usr/local/bin ioconfigInstall /usr/local/bin ioconfigInstall root root  555 command 30 % ioconfigInstall  >/dev/con 2>&1
%
#
```

And here is seen the same excerpt, but from a `transfer.ref` file generated after taking away the 'I' flag like is shown in the screencapture of Figure 12 Crate 'modules' configuration (after 'NextBlock').

```
#*****************************************************************************
# $Id: transfer.ref,v 1.40 2007/09/13 15:30:13 bracke Exp $
#*****************************************************************************
# WARNING: File generated from database. Can be overwritten at any time !
#

# ***** IOCONFIG Information *****

#   ln mln bus mtno module-type      lu W AM DPsz basaddr1   range1 W AM DPsz basaddr2   range2 testoff  sz sl ss
#+#  1   0 PCI  502 CTRP             0 N -- DP16       0       0 N -- ----       0       0       0  0  1  2
#+#  2   0 VME  186 RIO3-8064RD       0 N -- DP16       0       0 N -- ----       0       0       0  0  1 -1
#+#  3   0 VME   82 BNLDSP            0 Y EX DP32  2000000   e00000 N -- ----       0       0   600000  4  5 -1
#+#  4   0 VME   82 BNLDSP            1 Y EX DP32  3000000   e00000 N -- ----       0       0   600000  4  6 -1
#+#  5   0 VME   82 BNLDSP            2 Y EX DP32  4000000   e00000 N -- ----       0       0   600000  4  7 -1

#   ln sln bus mtno module-type      lu evno  subaddr    A1 F1     D1    A2 F2     D2
#
# ***** Program Startup before drivers *****
#
#
# start local errors output
#% prio 18   errlocal  >/dev/con 2>&1 &
#
#% cd /usr/local/drivers/sacvme; sacvmeinstall -R0 -M0 -V254 -L2
#% cd /usr/local/drivers/ctr; ctrinstall
#% cd /usr/local/drivers/BnldspVME; modinst  BNLDSP  -U0  -O2000000 -V220 -L2 -,  -U1  -O3000000 -V221 -
L2 -,  -U2  -O4000000 -V222 -L2
#
#% upfiles -all
#% wreboot -all
#% end
#
# ***** Program Startup after drivers *****
#
#
# Install data used by ioconfig library
/usr/local/bin ioconfigInstall /usr/local/bin ioconfigInstall root root  555 command 30 % ioconfigInstall  >/dev/con 2>&1
%
#
```

The bold printing lines are the result of taking out the 'I' flag for all 3 instances of ModuleType reference '`BNLDSP`'. The information for actually loading the driver in memory in the line: `cd /usr/local/drivers/BnldspVME; modinst BNLDSP –U0 –O2000000 –L2, -,` …. etc. comes from the expansion of the `BNLDSP` 'Module Type Reference' macro in the overview of

Figure 12 Crate 'modules' configuration (after 'NextBlock'). This expansion is shown in the database form like:

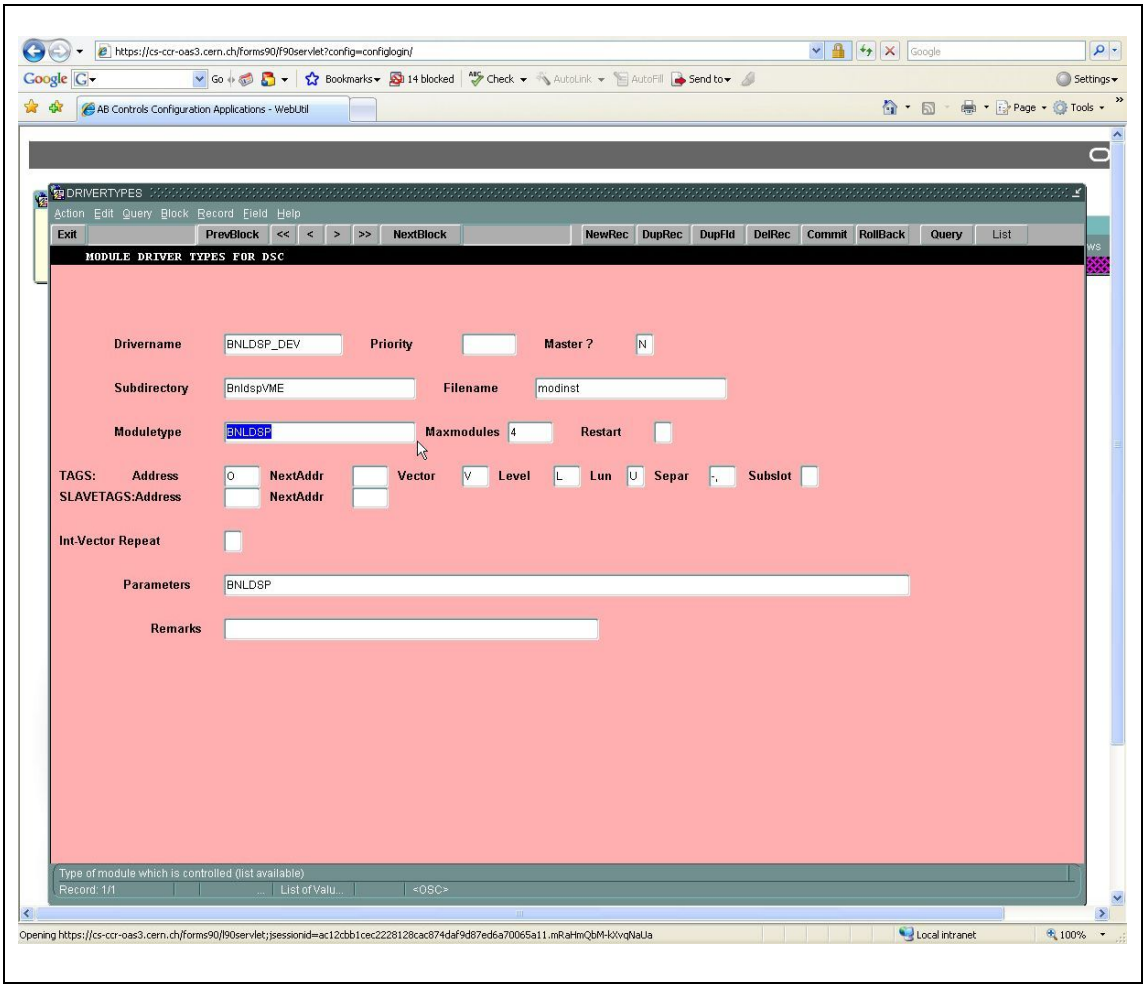

Figure 13 'Module Type Reference' expansion.

We come here via:

- Oracle hardware application, selecting menu '`Definitions`' dropdownlist and then '`# DSC List of drivertypes`'.
- Filling in the query form field '`Moduletype`' with '`BNLDSP`'.
- Click tab: 'Execute Query'.

I shall not go into more detail of this aspect of the hardware controls database usage. Some further information can be found in the document: 'Hardware configuration management for the DSC: A functional description.' written by Alain Gagnaire and which is also in the `DOCUMENTATION` directory of the Bnldsp C software at the CERN CVS repository.

You can also find it (with other related documents) at:

http://bracke.home.cern.ch/bracke/HTML/LEIR_Development/LEIR_Development.htm

---