TEMENOS™

TAFJ-DBTools

R14/R15

10/3/2015

Temenos

Amendment History:

Revision	Date Amended	Name	Description
1	25/08/2011	TAFJ team	Initial version
2	7st February 2012	H. Aubert	R12GA review
3	26 October 2012	JN. Charpin	Command execution report
4	15 January 2013	JN. Charpin	R13 review
5	17 February 2014	JN. Charpin	R14 review
6	10th April 2014	R. Vincent	Add ShowDicts documentation for JQL mode.
7	15th April 2014	H. Aubert	R14GA review
8	6th March 2015	H. Aubert	R15 AMR review
9	13th March 2015	G.Gowrimani	Promoted column inclusion
10	27th August 2015	R. Vincent	Promoted column example and further doc

Copyright

Errata and Comments

If you have any comments regarding this manual or wish to report any errors in the documentation, please document them and send them to the address below:
Technology Department

Temenos Headquarters SA
2 Rue de l'Ecole-de-Chimie,
CH - 1205 Geneva,
Switzerland

Tel SB: +41 (0) 22 708 1150
Fax: +41 (0) 22 708 1160

Please include your name, company, address, and telephone and fax numbers, and email address if applicable. TAFJdev@temenos.com

Table of Contents

Copyright....3
Errata and Comments....3
DBTools....7
Introduction....7
Syntax....8
Execution – interactive mode....8
Execution – script mode....9
Silent mode: no execution report....11
Presentation....12
Header....13
Main panel....13
Footer....15
Database Functionalities....15
SQL....15
JQL....16
OFS....18
DBCHECK....19
JED....20
Extract table....21
Load data....23
Show Locks....24
Release Locks....26
Database statistics....27
Console Functionalities....28
HELP....28
History....29
Aliases....30
Create an alias....30
Display aliases....31
Execute an alias....31

Remove an alias....33
Setup....34
Max row retrieved (SQL)....34
Max records per page....34
Max line length....35
History size max....35
Show session long name....35
Show page number....35
SQL auto-commit....35
Log file....35
Navigation and display....36
Change page/ column....36
Change console display....36
JED Console application....37
Syntax....37
Execution – Interactive mode....37
Optional parameters....37
Execution - script mode....38
Presentation....40
Header....40
Main panel....42
Footer....43
Functionalities....43
Change a record....43
Delete a record....52
Validation....53
Navigation and display....56
DBCheck....65
Introduction....65
Syntax....65
Example....65
DBComparer....66
Introduction....66
Requirements....66

Syntax....66
Example....67
TAFJ Promoted Columns....68
Promoting Columns for SQLServer example....68
Promoting Columns for DB2 example....68
Promoting Columns for Oracle example....68
Drop and Recreate the Views (All Databases)....69
USE CASE....71
Alter Table for each multi-value....71
Create Indexes for each new column....71
Recreate the view with new columns....71
Change query....72

DBTools

Introduction

TAFJ-DBTools Console provides access to SQL / JQL / OFS statements, and integrates various TAFJ tools that deal with database within a unique application.

It is also designed to help users to launch various commands in a quick and efficient way by providing aliases and history functionalities. Results are being displayed with a customizable and friendly approach.

The available functionalities are:

- Execute SQL statements
- Execute JQL statements
- Execute OFS statements
- Execute JED application (edit and modify record)
- Extract data from database to file
- Load data from file to database
- Show current locks on database
- Release locks on database
- Display statistics on table

Syntax

Execution – interactive mode

DBTools Console could be executed from TAFJ_HOME/bin.

```
DBTools [-cf <configuration>[.properties]]
```

The targeted database is the one defined in the default configuration file or in the configuration file specified with the -cf option.

Note:

Copy all jar files from ***TAFJ_HOME\dbdrivers\<drivers for different databases>*** folder to ***TAFJ_HOME\ext folder.***

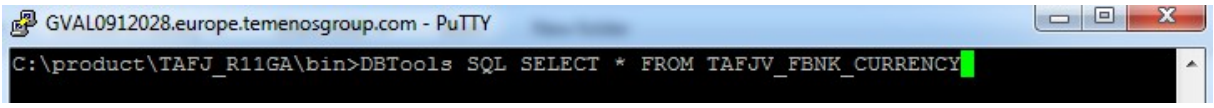
DBTools Console gives also the ability to launch automatically a default command.

Just add your command to the command line after DBTools Console invocation and it will be executed by default.

```
DBTools [-cf <configuration>[.properties]]  COMMAND
```

i.e.

To launch a SQL command as default command when executing DBTools Console.



Execution – script mode

DBTools Console could also execute command in script mode without opening the console. You just have to add '-s' option and the command you want to execute.

```
DBTools [-s [command] ]
```

i.e.

To script a JQL command

To script a SQL command

Please note to script a "SELECT * FROM …" command you will have to escape the * parameter

DBTools -s SQL SELECT * FROM FBNK_CURRENCY

Command execution report

When executing a command DBTools could produce different execution reports.

Default mode: execution report buffered in DBTools console

By default without any parameter specified, DBTools console is opened, the command is executed and result is appended to a buffer and displayed in the console at the end of the execution.

Script mode: execution report appended live in OS console

As mentioned above, when providing **-s** as parameter, "script mode" is enabled, DBTools console is not opened during execution. The execution report is appended live in the OS console.

Log mode: execution report appended in a log file

When providing **-log logFileName** execution report is redirected live to the related log file. This file is generated with .log extension under TAFJ_HOME/log

The “log mode” could be associated to the “script mode” by combining -s and -log logFileName parameters.

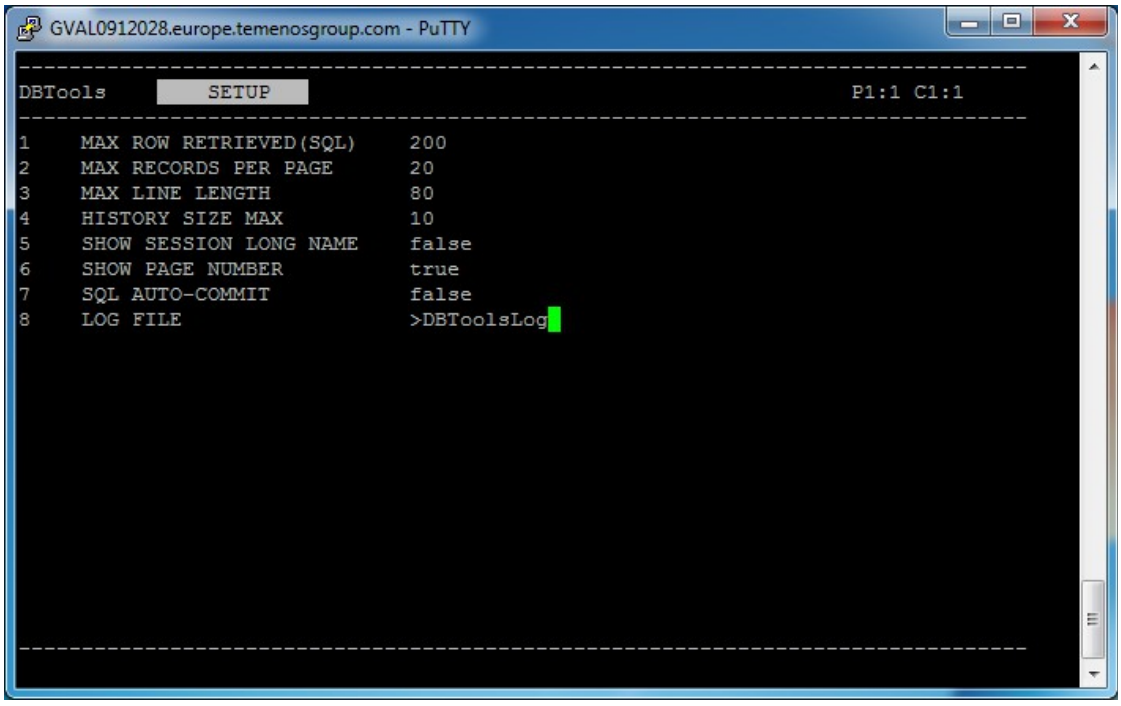
i.e.

```
DBTools -s -log currencyId SQL SELECT RECID FROM FBNK_CURRENCY
```

In that case all FBNK_CURRENCY IDs will be outputted to the OS console and to currencyId.log under TAFJ_HOME/log.

The “log mode” could be associated to the “console mode” by providing a log file name in the DBTools console, from setup menu.

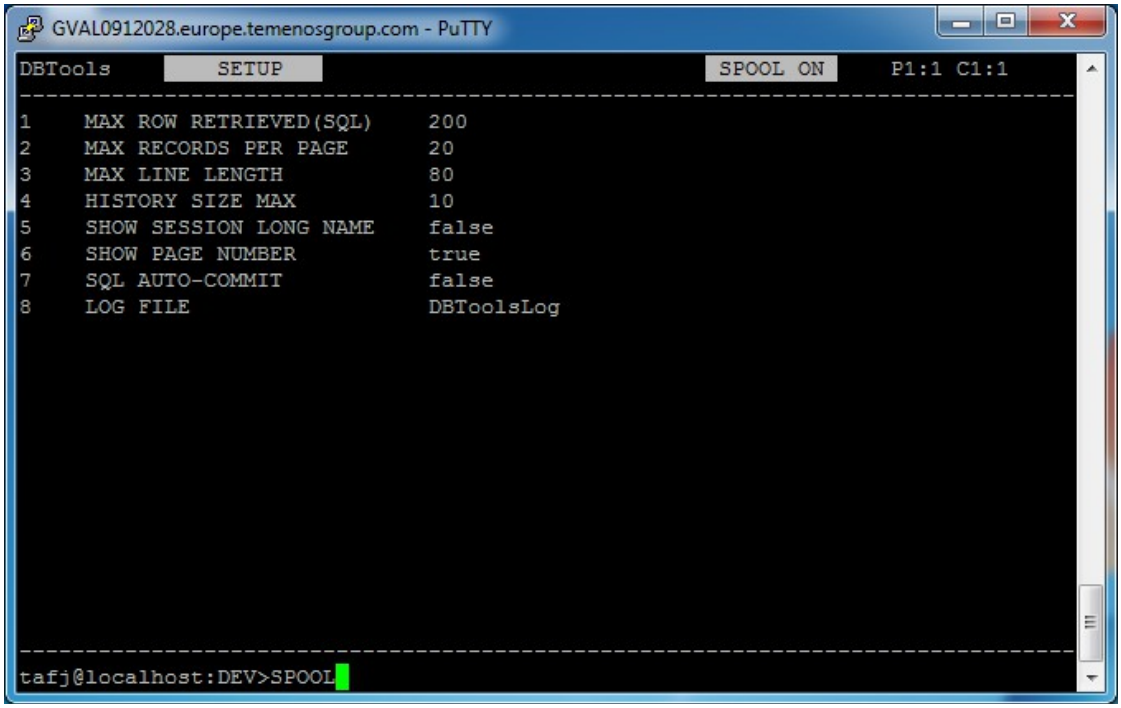
The setup menu is available by typing **setup** on the command line. Please refer to the setup section for more information about it.



Once the log file has been setup it’s necessary to turn on the **SPOOL** feature to tell the console to output the result to the log file.

It could be done by typing **SPOOL** on the command line. You will see that SPOOL is ON in the header of the console. By retyping **SPOOL** you will turn it off and disable the logging functionality.

This way you could choose which command execution report you want to append to log file.



Silent mode: no execution report

When providing parameter **-silent** in “script mode”, command is executed silently, no output is generated.

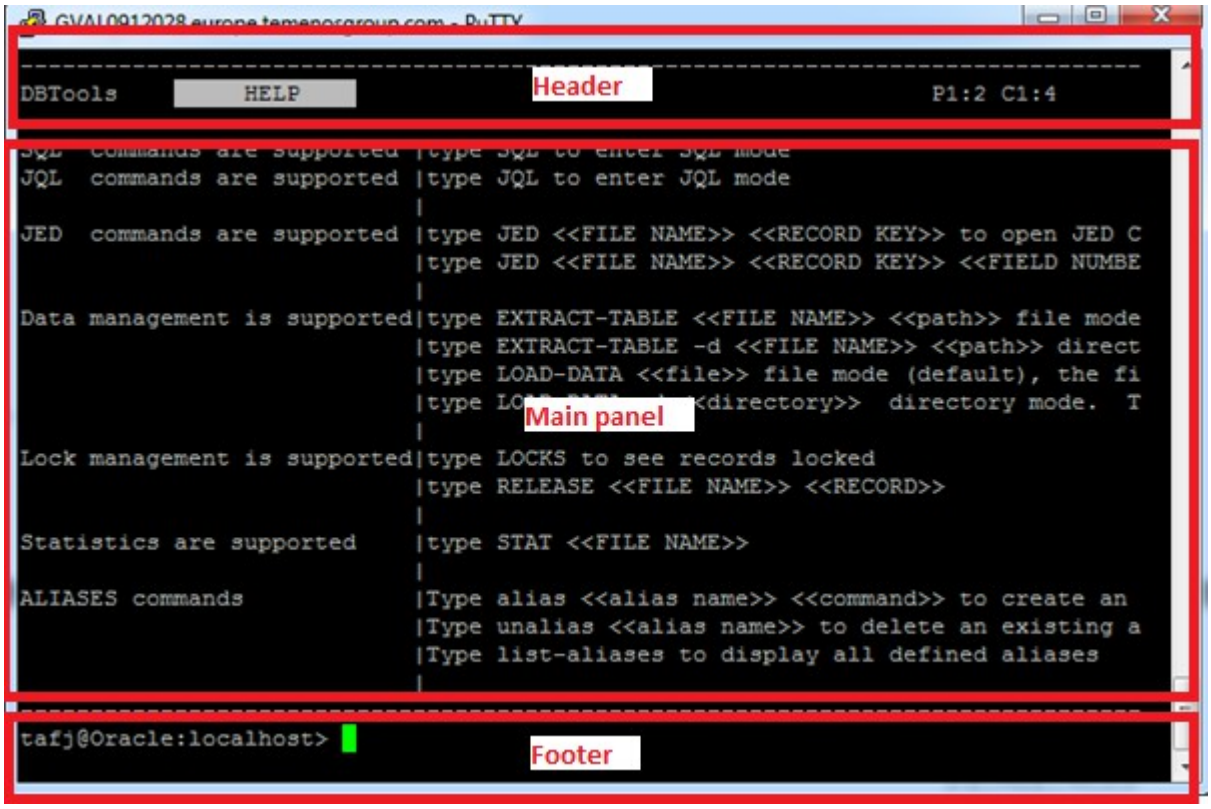
i.e.

```
DBTools -s -silent COMMAND
```

Presentation

DBTools console consists of three panels:

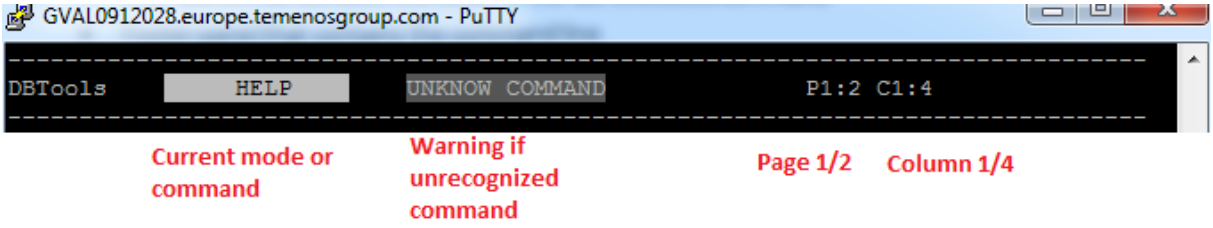
- Header panel that contains information about the current mode or running command
- Main panel that contains the result of the last executed command
- Footer panel that contains the command line



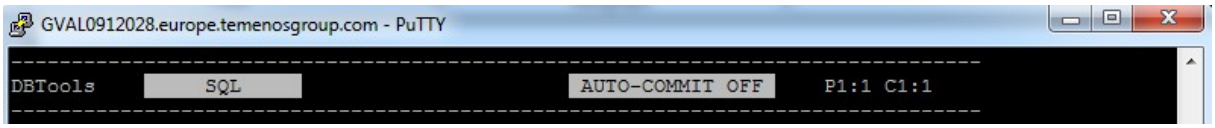
Header

The header presents the following information:

- The current mode or running command
- A warning in case of unrecognized command
- The current page / column number and the total number of page / column if the result of the command is being displayed on multiple page and/or columns.



In SQL Mode, you will also see the auto-commit status. Please refer to SQL and setup sections to get more information on that point.



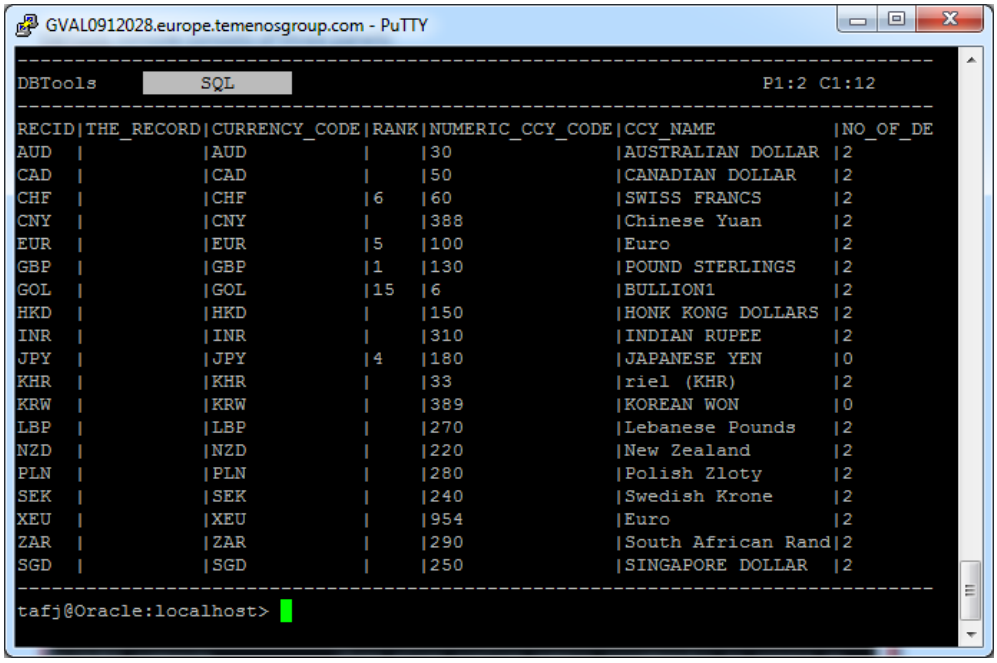
Main panel

The main panel contains the command result to be displayed or edited. It is displayed vertically and split into lines and eventually columns.

If the data is larger than the console maximal length (80 characters by default) it will be displayed on multiple columns.

If the result has more lines than the console could support (20 rows by default) it will be displayed on multiple pages.

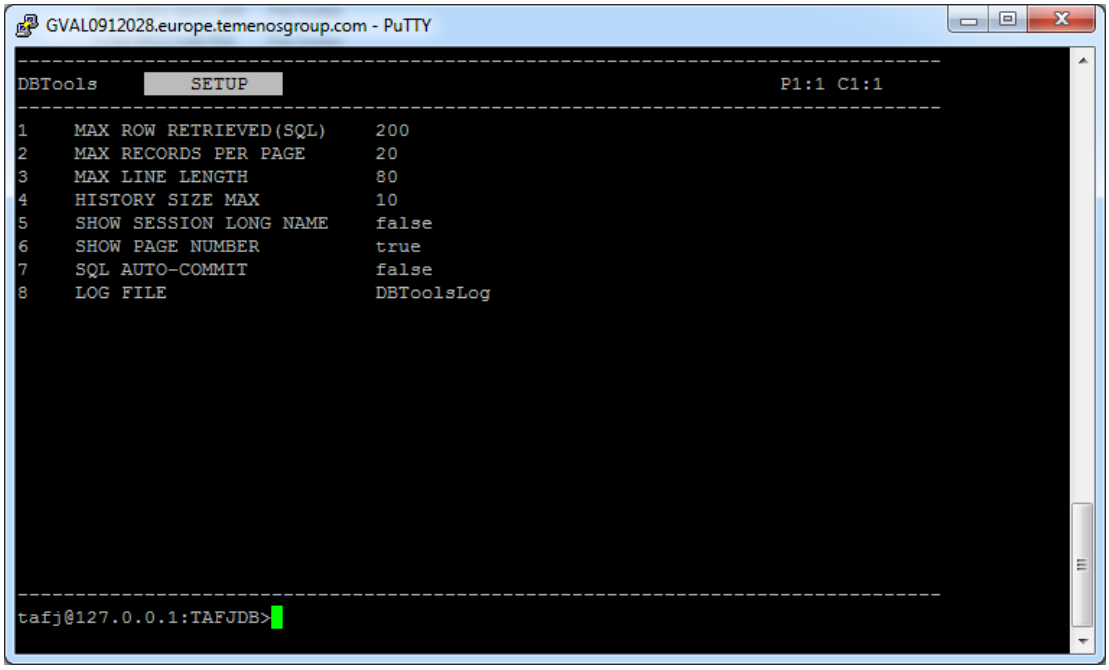
i.e. SQL Select result that is being displayed on 2 pages and 12 columns.



For command results that can be edited, i.e. console setup parameters, a field number is displayed on a margin on the left part of the panel; the associated value is displayed next to the margin on the middle of the panel.

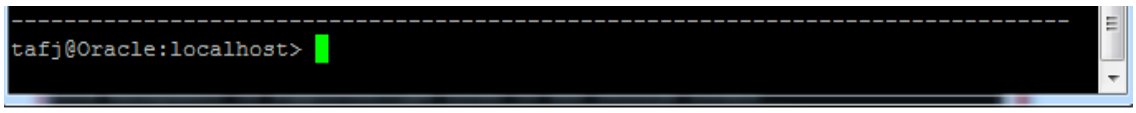
It's then possible to access the related data by typing its associated field number on the command line. Please refer to Setup functionality section for more information.

i.e. Setup command result that can be edited, by typing '2' you will be able to change actual value '20' to another value.

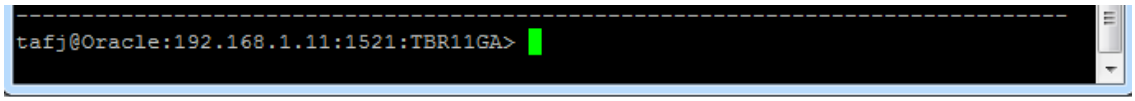


Footer

The footer panel contains the command line that shows by default the database connection parameters: user name, database provider and hostname.



Depending on console setting it could also show the hostname ip address and the database name.

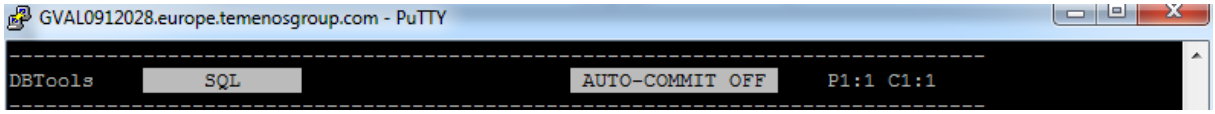


Database Functionalities

SQL

To execute SQL statements, first you have to enter **SQL mode** by typing command "**SQL**" on the command line or **by prefixing your SQL command with SQL keyword**.

Then you will see that SQL mode is activated in the header panel and the auto-commit status.



The console is now ready to execute SQL statements.

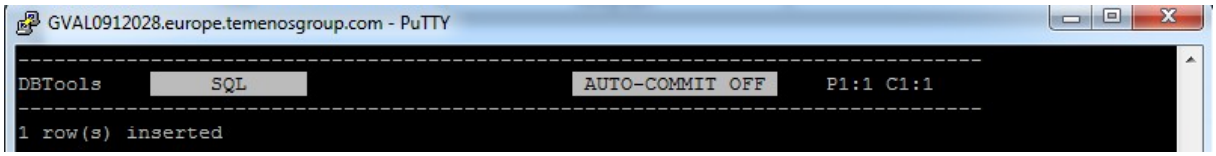
To be recognized the statement must start with one of the following keywords in uppercase:

- SELECT
- INSERT
- UPDATE
- DELETE

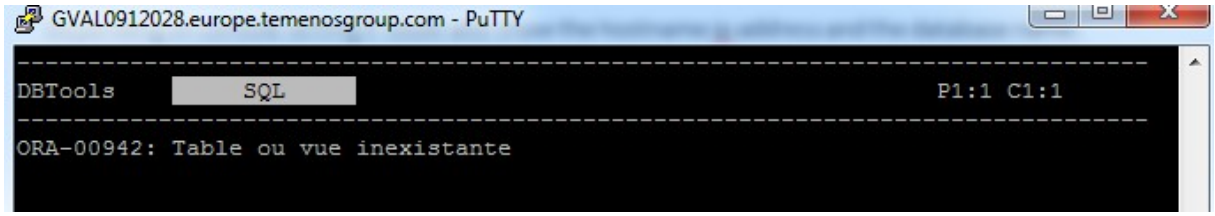
- CREATE
- ALTER
- DROP
- COMMIT
- ROLLBACK
- DESCRIBE

By default the database connection is set to auto-commit=false, that means if you don't apply the COMMIT statement after your data modification, your change will be roll backed when exiting the console.

Once your statement has been executed, the result will be displayed on the console if that statement has got a result set, otherwise you will get information about the number of rows affected.



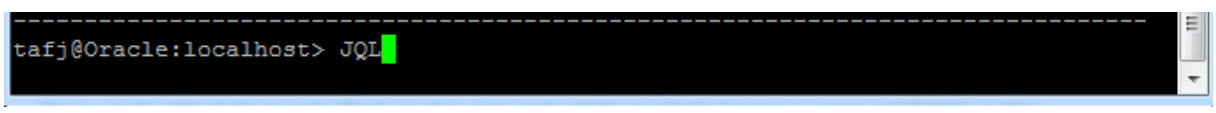
In case of SQL Exception the related message will be displayed on the console.



To avoid an excessive memory usage when retrieving important set of data, by default only the first 200 rows will be displayed. You can change this parameter to the value you want, please refer to Setup section to do so.

JQL

To execute JQL statements, first you have to **enter JQL mode** by typing command "**JQL**" on the command line or by **prefixing your JQL command with JQL keyword**.



Then you will see that JQL mode is activated in the header panel.

The console is now ready to execute JQL statements.

To be recognized the statement must start with one of the following keywords in uppercase:

- COPY <<Source Table Name>> <<Destination Table Name>> <<RecId optional>>
- COUNT <<VOC File name>>
- GETDBNAME <<VOC File name>>
- LIST-ITEM <<VOC File name>> <<WITH clause>>
- DESCRIBE <<View or Table Name Pattern>> <<Column Name Pattern (optional)>>...ie. TAFJV_FBNK_CURR% %CURR%
- LOCATE-TABLE <<Table Name>> (zOS only)
- CLEAR-FILE <<VOC File name>>
- CREATE-FILE <<VOC File name>>
- DELETE-FILE <<VOC File name>>
- SHOWDICTS <<TYPE= (I|D|PH>> <<ATTRIBUTE= (ALL| Attribute Number to Search>> <<(STARTSWITH|ENDSWITH|CONTAINS)="Search String">> <<FILE= FileName to log.>>

SHOWDICTS allows you to search all dictionary items. Logs show up in $TAFJ_HOME/log. If the FILE parameter is not specified, the default ShowDicts.txt is used.

This example below shows any dict item I type with ATTRIBUTE 2 containing the search “ENQ.TRANS”:

DBTools>SHOWDICTS TYPE=I ATTRIBUTE=2 CONTAINS="ENQ.TRANS" FILE=enqtrans.txt

The output file $TAFJ_HOME/log/enqtrans.txt would look like:

```
FMF1.MD.DEAL FMF1_MD_DEAL D_F_MD_DEAL  CUST.SECTOR  I?CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "SECTOR")??CUST.SECTOR?4R?S????????CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "SECTOR")
```

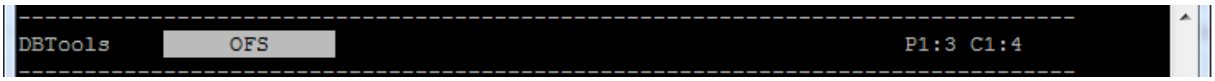
```
FMF1.MD.DEAL FMF1_MD_DEAL D_F_MD_DEAL  INDUSTRY     I?CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "INDUSTRY")??INDUSTRY?4R?S????????CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "INDUSTRY")

FMF1.MD.DEAL FMF1_MD_DEAL D_F_MD_DEAL  RESIDENCE    I?CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "RESIDENCE")??RESIDENCE?2L?S????????CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "RESIDENCE")

FMF1.MD.DEAL FMF1_MD_DEAL D_F_MD_DEAL  SECTOR I?CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "SECTOR")??SECTOR?4R?S????????CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "SECTOR")

FMF1.MD.DEAL FMF1_MD_DEAL D_F_MD_DEAL  TARGET I?CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "TARGET")??TARGET?4R?S????????CUSTOMER;
SUBR("ENQ.TRANS","CUSTOMER", @1, "TARGET")
```

This format is easily imported into Excel with tab delimiters.

OFS

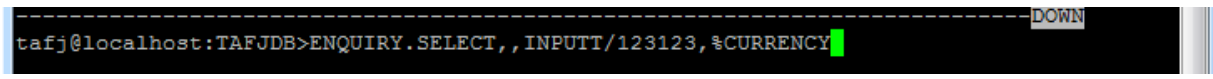
To execute OFS statements, first you have to **enter OFS mode** by typing command "**OFS**" on the command line or by **prefixing your OFS command with OFS keyword**.

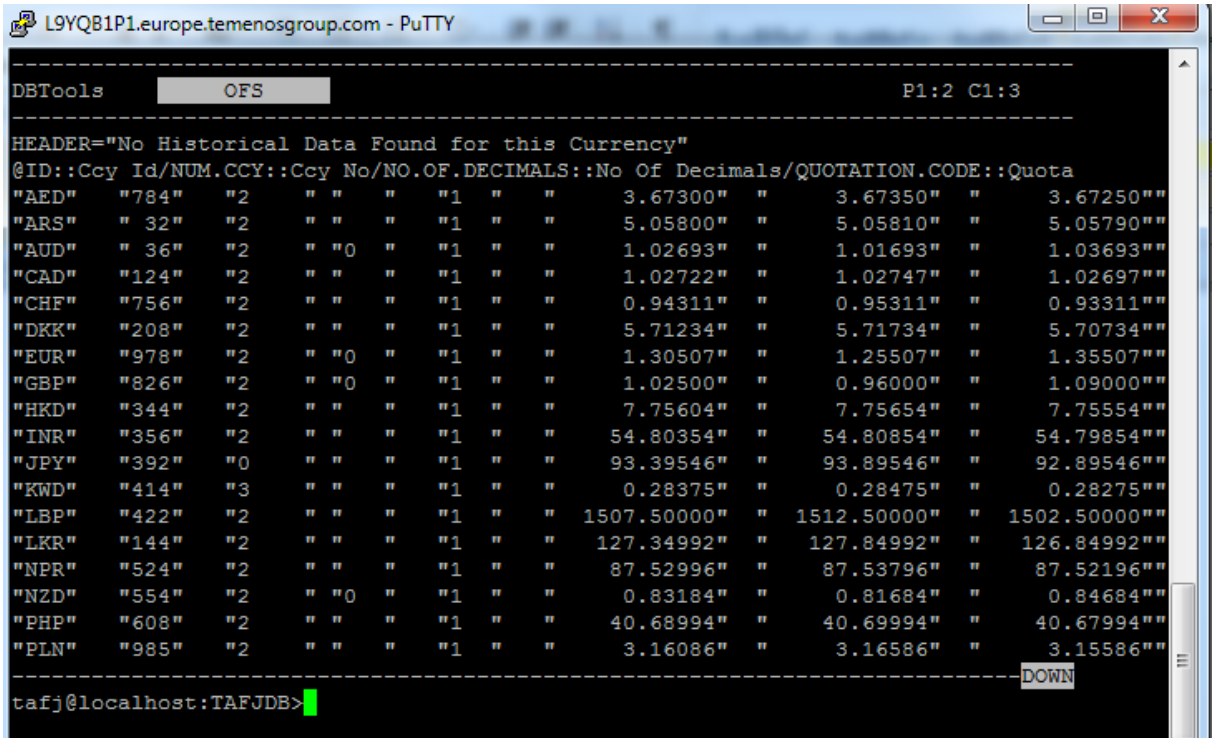
Then you will see that OFS mode is activated in the header panel.



The console is now ready to execute OFS statements, please note that your TAFJ properties file must point to a valid T24 precompile directory or jars.

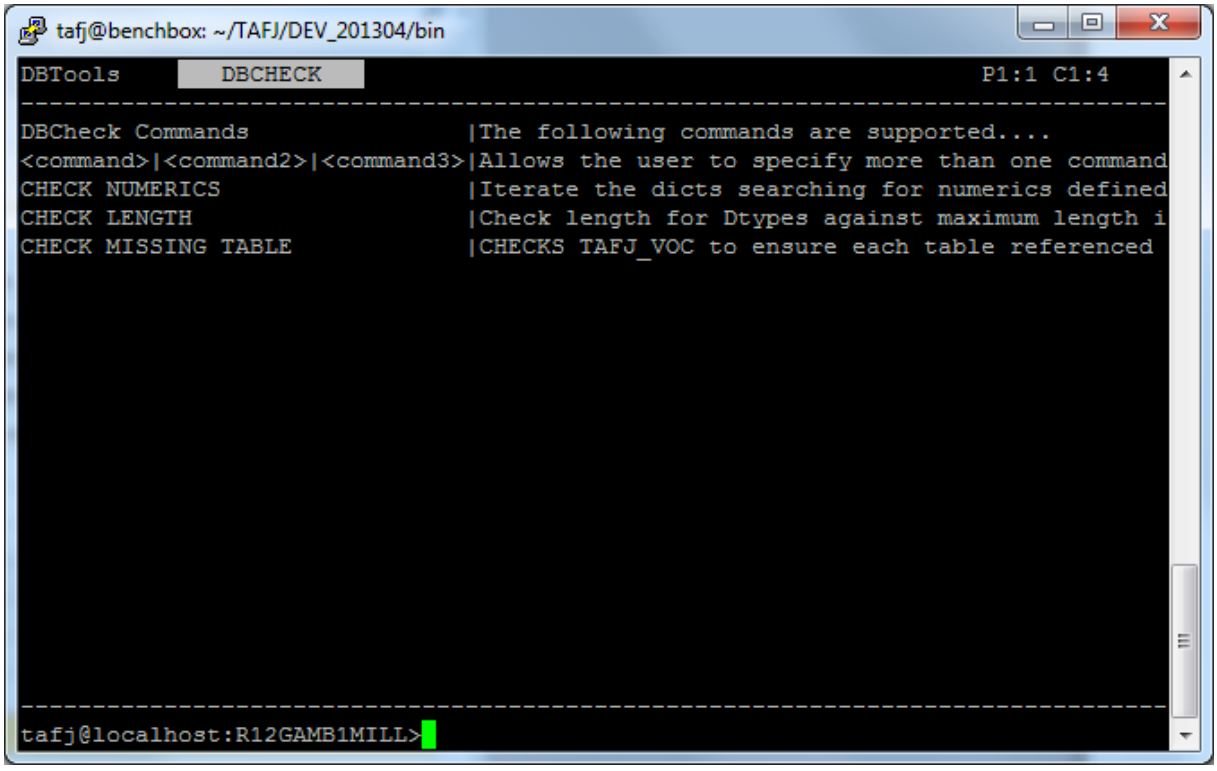
The default OFS source is GCS.




L9YQB1P1.europe.temenosgroup.com - PuTTY
DBTools OFS P1:2 C1:3
HEADER="No Historical Data Found for this Currency"
@ID::Ccy Id/NUM.CCY::Ccy No/NO.OF.DECIMALS::No Of Decimals/QUOTATION.CODE::Quota
"AED" "784" "2 " " " "1 " " 3.67300" " 3.67350" " 3.67250""
"ARS" " 32" "2 " " " "1 " " 5.05800" " 5.05810" " 5.05790""
"AUD" " 36" "2 " "0 " "1 " " 1.02693" " 1.01693" " 1.03693""
"CAD" "124" "2 " " " "1 " " 1.02722" " 1.02747" " 1.02697""
"CHF" "756" "2 " " " "1 " " 0.94311" " 0.95311" " 0.93311""
"DKK" "208" "2 " " " "1 " " 5.71234" " 5.71734" " 5.70734""
"EUR" "978" "2 " "0 " "1 " " 1.30507" " 1.25507" " 1.35507""
"GBP" "826" "2 " "0 " "1 " " 1.02500" " 0.96000" " 1.09000""
"HKD" "344" "2 " " " "1 " " 7.75604" " 7.75654" " 7.75554""
"INR" "356" "2 " " " "1 " " 54.80354" " 54.80854" " 54.79854""
"JPY" "392" "0 " " " "1 " " 93.39546" " 93.89546" " 92.89546""
"KWD" "414" "3 " " " "1 " " 0.28375" " 0.28475" " 0.28275""
"LBP" "422" "2 " " " "1 " " 1507.50000" " 1512.50000" " 1502.50000""
"LKR" "144" "2 " " " "1 " " 127.34992" " 127.84992" " 126.84992""
"NPR" "524" "2 " " " "1 " " 87.52996" " 87.53796" " 87.52196""
"NZD" "554" "2 " "0 " "1 " " 0.83184" " 0.81684" " 0.84684""
"PHP" "608" "2 " " " "1 " " 40.68994" " 40.69994" " 40.67994""
"PLN" "985" "2 " " " "1 " " 3.16086" " 3.16586" " 3.15586""
DOWN
tafj@localhost:TAFJDB>

DBCHECK

You can execute the following commands to do checks on your database by typing DBCHECK to put DBTools in DBCHECK mode, then USAGE to arrive at the below screen:



All commands log to a particular file that is shown to the user once the command runs.

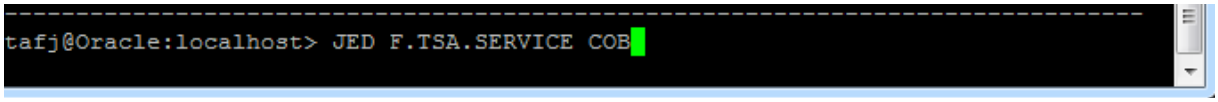
CHECK NUMERICS iterates the dictionary items where dictionary items are defined as numeric. It then tests those fields to make sure the data is indeed numeric. As well, it tests those VOC items that are defined rightJustified to see if the data is numeric.

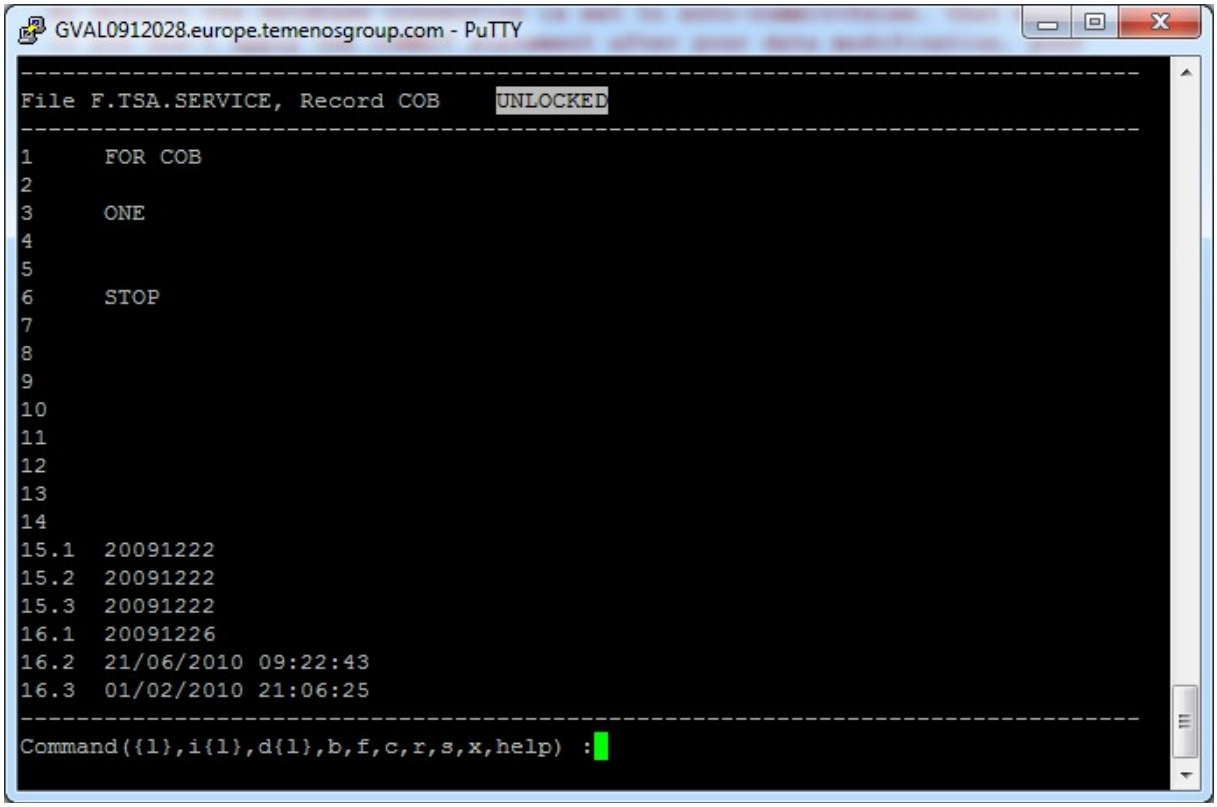
CHECK LENGTH iterates the dictionary items and gathers defined lengths and checks each against the maximum length stored in the column. It reports anything that is incorrectly defined. THIS COMMAND TAKES A LONG TIME TO RUN (More than 3 hours depending on the size of the DB).

CHECK MISSING TABLE iterates VOC items and tests that the tables exist in the database.

JED

You can call JED console to display and edit database records by using command JED <<FILE NAME>> <<RECORD KEY>>





You can also call JED **-s** <<FILE NAME>> <<RECORD KEY>> <<FIELD NUMBER>>=<<VALUE>>; to directly update a record without opening JED Console.

For multiple attributes, separate them with ";".

Depending on the operating system you could have to escape the part of command after -s option with quotes: JED -s 'F.TSA.SERVICE COB 6=START;'

Please refer to the JED Console section at the end of this document to get more information about JED Console usage.

Extract table

This command allows extracting the records from a table and writing them on the local file system. Depending on the option you choose, the result will be:

-f : file mode (default). A set of distinct file, one per record, the name of the file has the format <filename>-<recordId>

-d : directory mode. A directory having the name of the table and inside this directory each record will be stored in a distinct file, the name of this last file will be the actual record Id.

If any file or directory already exists, they will be simply overridden.

To extract the CURRENCY table as a set of records, just type

This will result in a directory structure like this one:

/home/user/currency

FBNK.CURRENCY -GBP

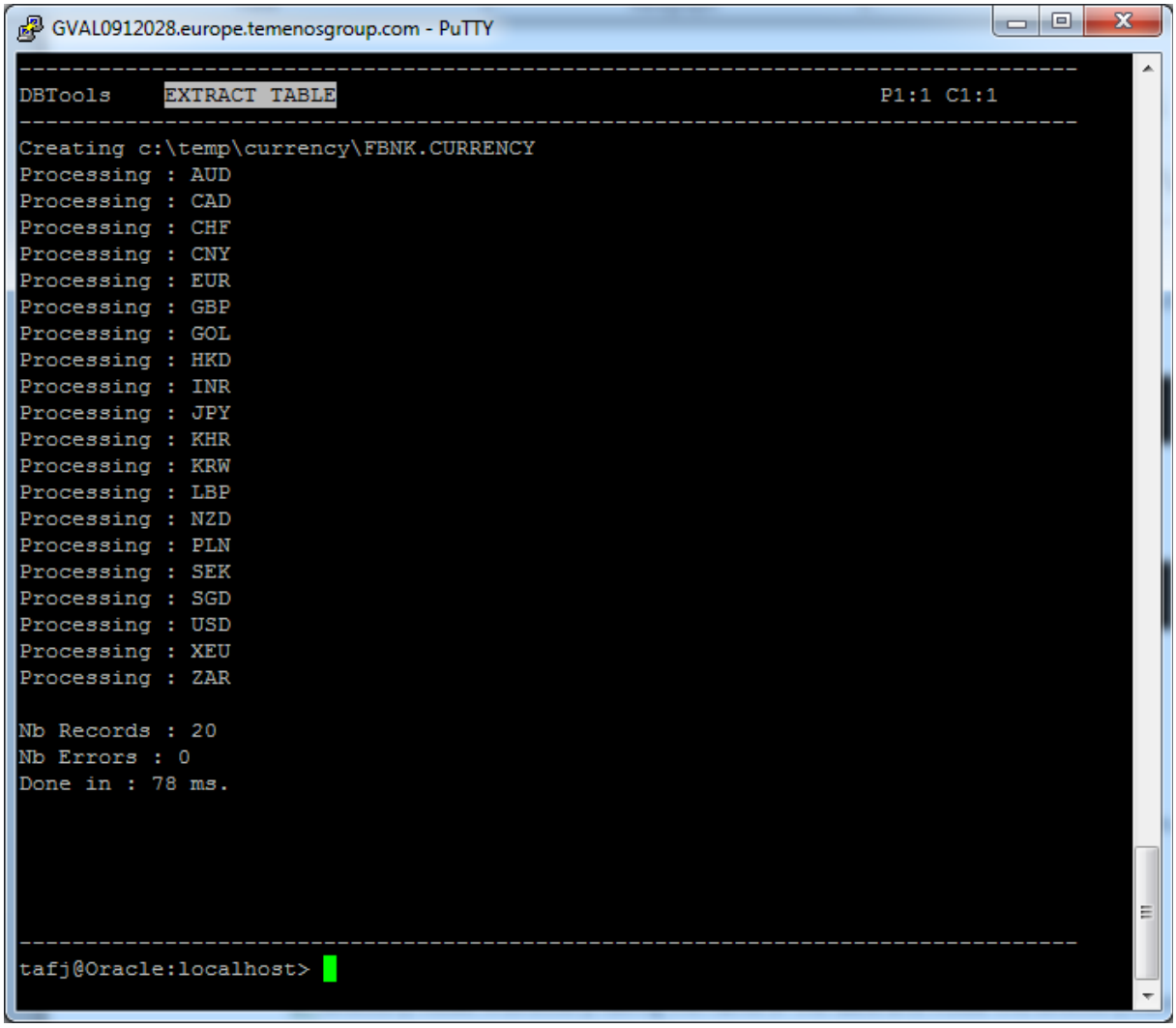
FBNK.CURRENCY -CHF

FBNK.CURRENCY -USD

. . .

Where FBNK.CURRENCY -GBP, FBNK.CURRENCY -CHF ... are files containing the record.

All extracted files extracted and eventually raised exceptions will be shown in the console as result.



To extract the CURRENCY table as a set of records in their own directory, just type

This will result in a directory structure like this one:

/home/user/

FBNK.CURRENCY/

GBP

CHF

USD

. . .

Where GBP, CHF ... are files containing the record.

All extracted files extracted and eventually raised exceptions will be shown in the console as result.

Known issue: as it is possible that the record id of the record to be extracted contain the character "/", extraction will fail as we can't have "/" in a file name.

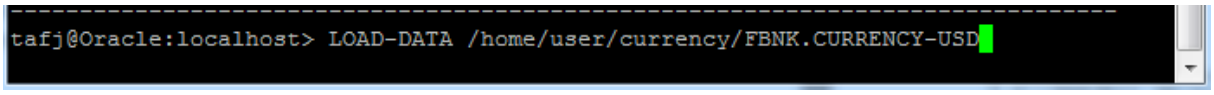
Load data

This command loads data from your file system into your database. You can provide as parameter a file to load a single record, or a directory to load a set of records i.e.:

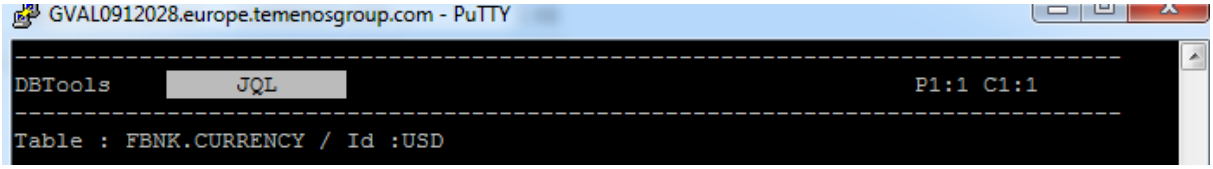
-f : file mode (default) . The file name has the format <filename>-<recordId>
If the file passed is a directory, all files having the previous format will be processed

-d : directory mode. The parameter must be a directory.
This directory will be recursively processed until a file is found.
The parent directory of this file represents the FileName, file name itself is the recordsId

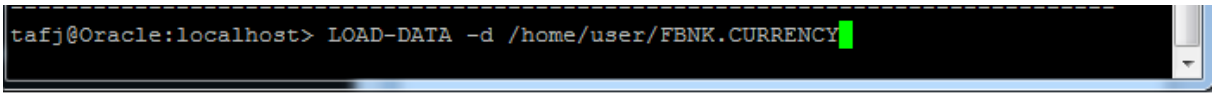
To load a single record from a file on your file system

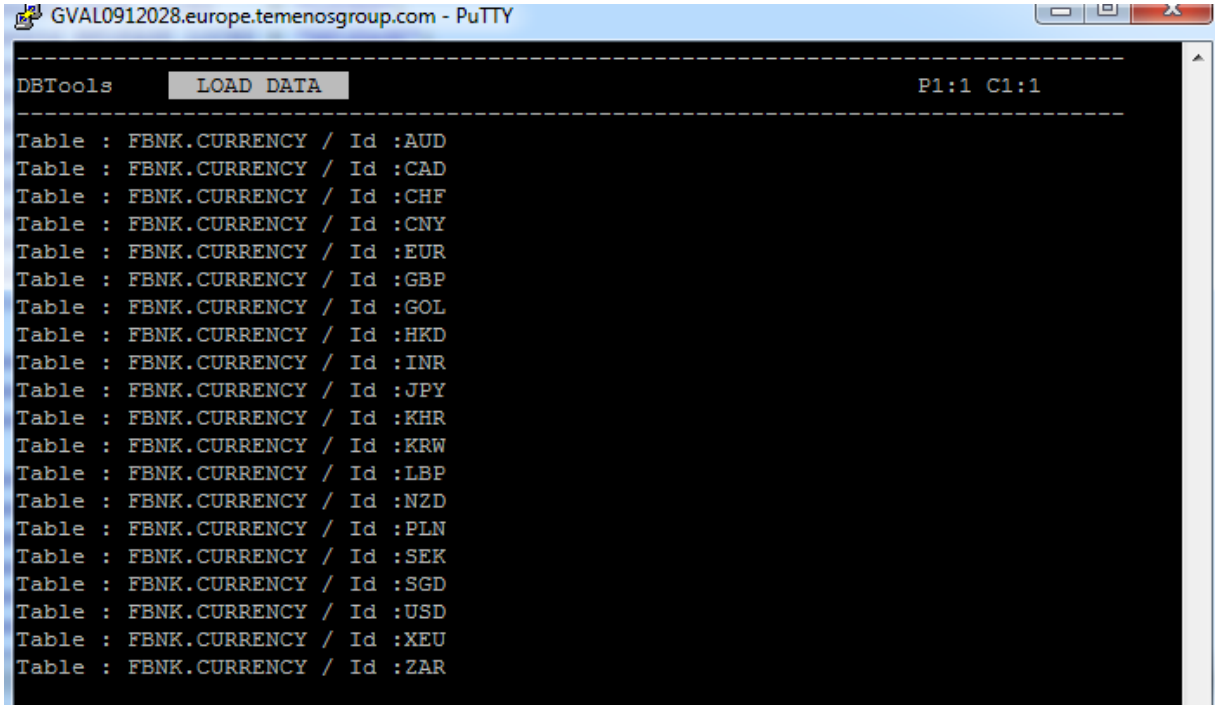


You will get as result displayed in the console the table impacted and the record id inserted.



To load a set of records from a directory on your file system

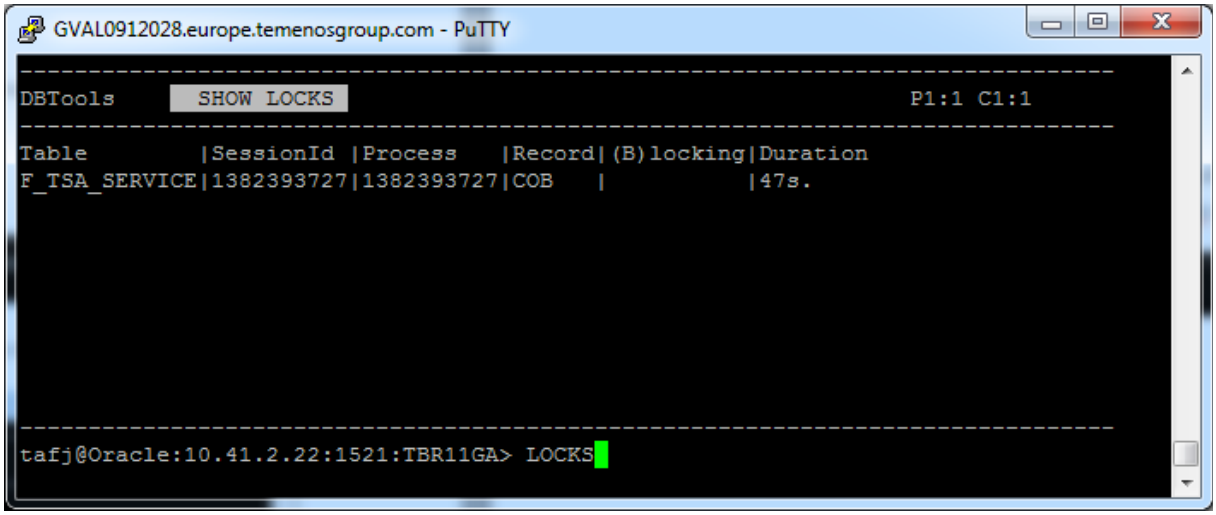




Show Locks

By launching LOCKS command you will be able to see all records locked on the database. It monitors locks managed by tLockManager, the TAFJ LOCK_RECORDS table (JDBC Locking), or ORCL, or MSQL, or DATABASE internal locking on the database. Below is a screenshot from tLockManager.

The result displayed is the following.



This is different than what you might see with DATABASE locking. When issuing the LOCKS command, non-existent rows will appear as NonExist*<<TAFJ Session Unique ID>> for the Session Id column, whereas one that was taken via SELECT FOR UPDATE, will show the Oracle Session ID for the Session Id column. One cannot release locks taken with SELECT FOR UPDATE. The Oracle session must be either committed, rolled back, or killed. As well, tables with more than 25,000 rows will show up as F_PGM_DATA_CONTROL *UNKNOWN, if F_PGM_DATA_CONTROL had more than 25,000 rows. This is because Oracle has no way to report row level locking except by iterating all of the rows and attempting to lock them. This is very costly in terms of performance, and hence, why TAFJ caps the number of rows.

DBTools SHOW LOCKS P1:1 C1:1

LOCK ID |Session Id

FBNK_CURRENCY*ZZZ |NonExist*1193893622

FBNK_CATEG_ENTRY*169376209561194.050001|128

This is also different than what you might see with ORCL, or MSSQL locking. Below is a screenshot of ORCL locking. The 53 below denotes the *database* session id that *called* the lock. 205121125 represents the TAFJ unique session for a thread. So now TAFJ can match a java thread with the database session that locked it.

```
--------------------------------------------------------------------------------
-DOWN
tafjuser@10.41.5.54:TESTDB>LOCKS
--------------------------------------------------------------------------------
-----
DBTools      SHOW LOCKS                                                  P1:1
C1:1
--------------------------------------------------------------------------------
-----
LOCK ID          |Session Id
FBNK_CURRENCY*USD|2051521125*53
```

Keep in mind that there is one connection for a thread actually asking for a lock, and another connection that actually takes the lock (this is due to auto-commiting of the lock connection). What is reported is the database session id for the lock connection which will also appear in the locking.log (for JDBC Locking it will only appear in the log).

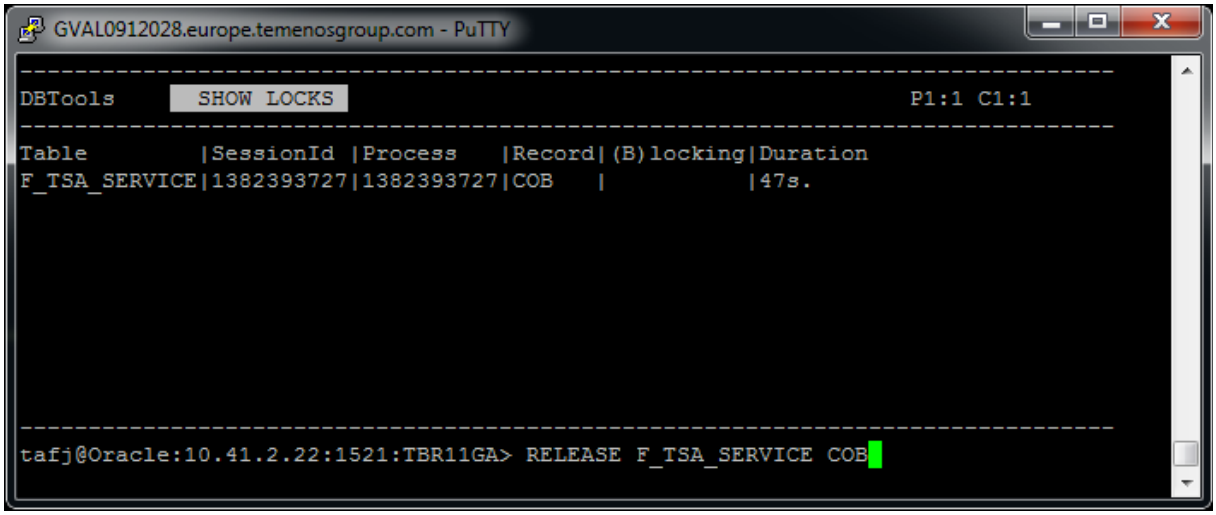
[DEBUG] 2015-04-02 15:18:40,656 [main] LOCKING - LOCKING FBNK_CURRENCY*USD / 2051521125*53...

[DEBUG] 2015-04-02 15:18:40,661 [main] LOCKING - LOCK FBNK_CURRENCY*USD / 2051521125*53 -> OK

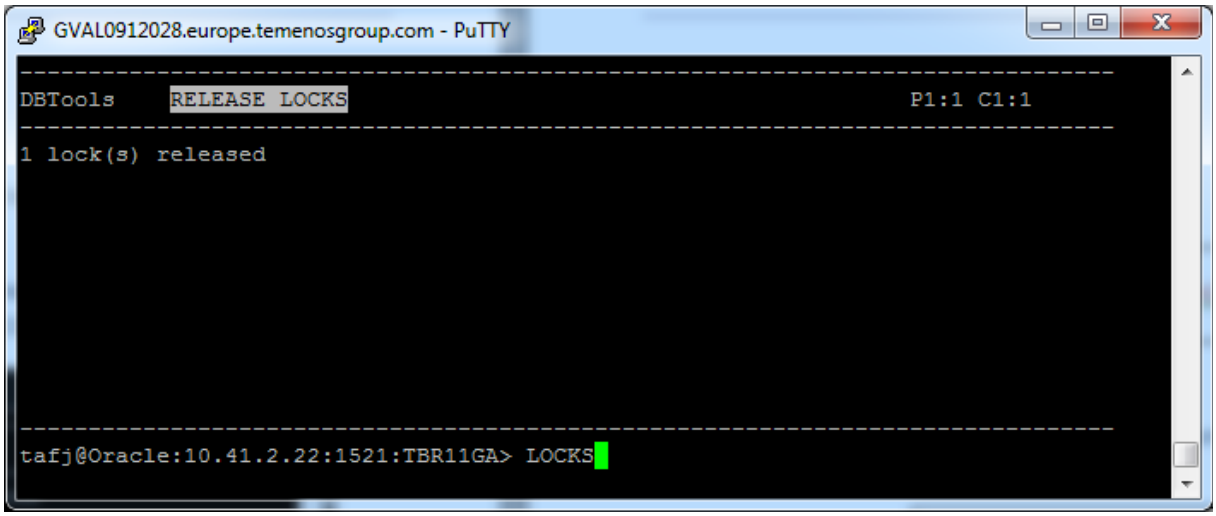
Release Locks

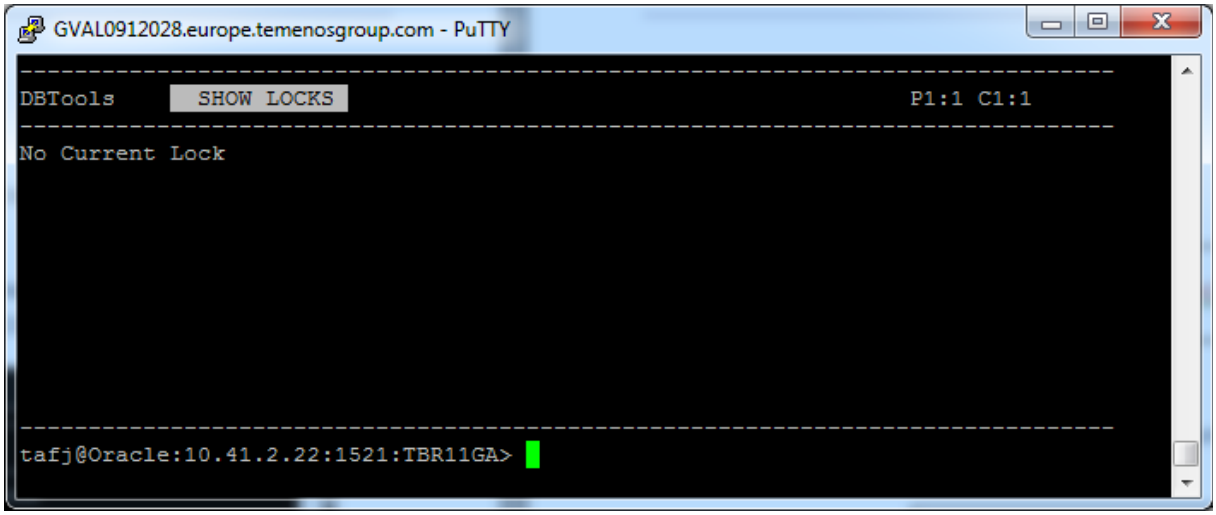
The release locks command is specially designed for tLockManager. It is used to release the Locked record from tLockManager.

You can check which records are currently locked by launching command LOCKS and then release the record you want by typing command RELEASE <<FILE NAME>> <<RECORD ID>>



Result being displayed after releasing the lock.





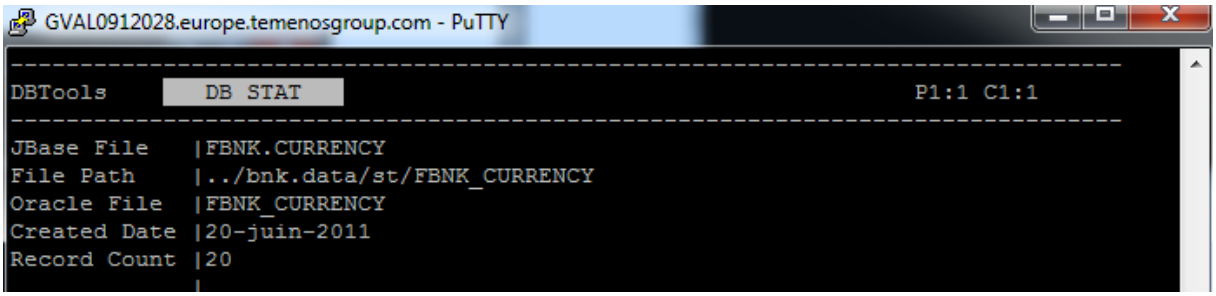
Note: With DATABASE locking, it is not possible to release locks that have been taken with SELECT FOR UPDATE (see above SHOW LOCKS section for more details)

Database statistics

This command allows to easily getting information about your JBase file, JBase file path, JBase equivalent database filename, created date and record count.

To get information about the file you have configured type STAT <<FILE.NAME>>

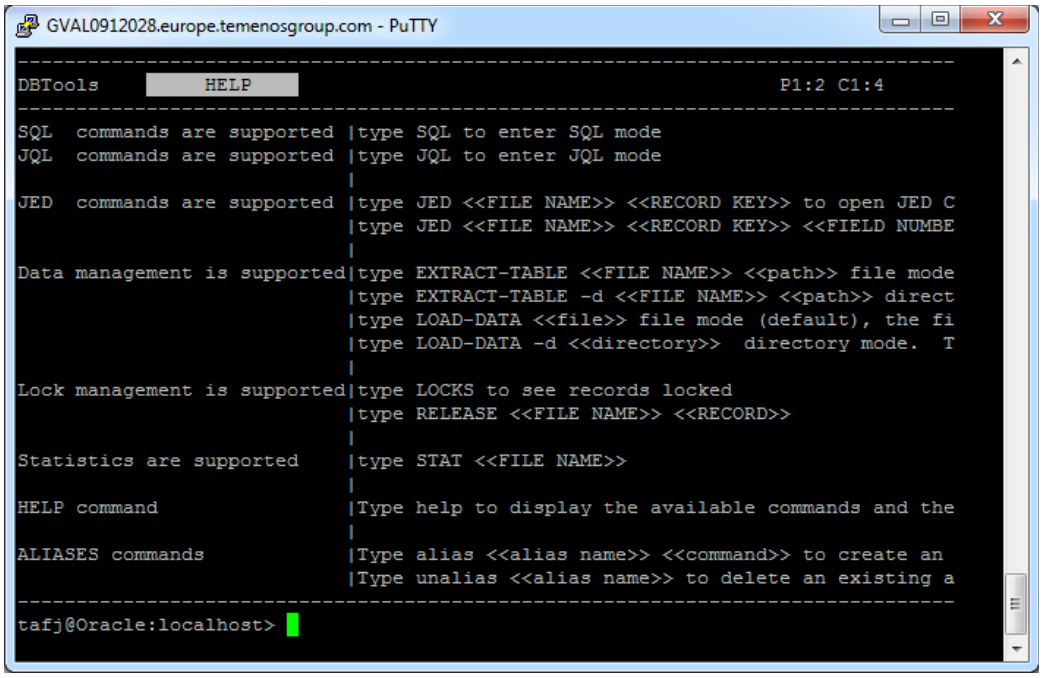
The resulting output looks like this:

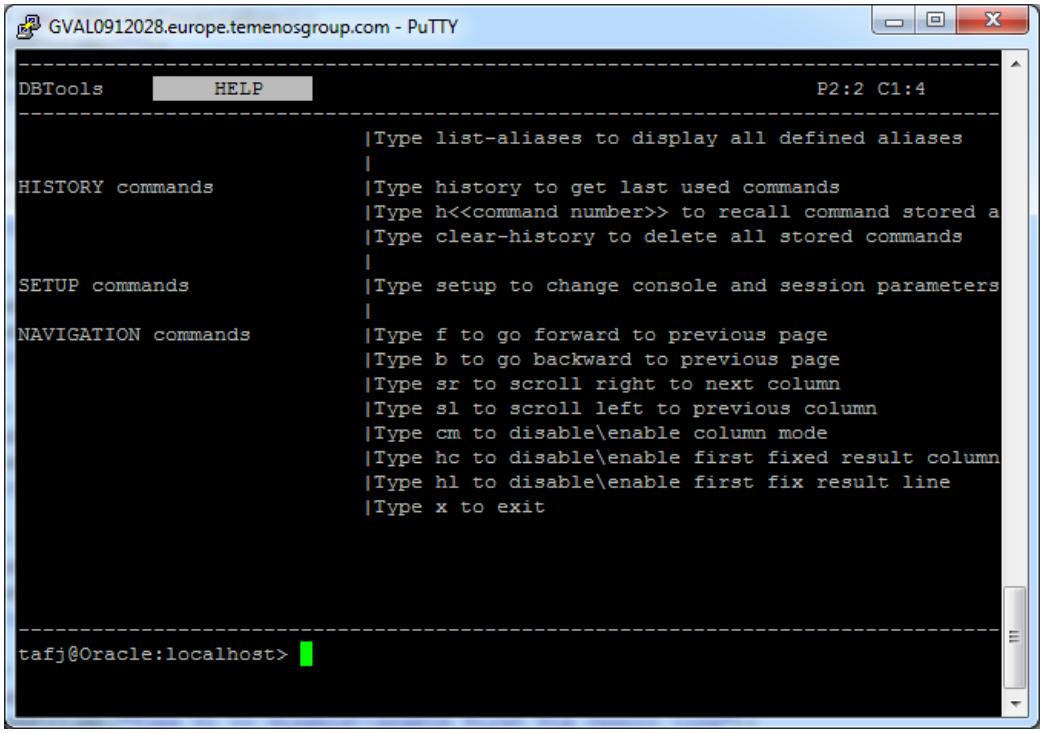


Console Functionalities

HELP

When opening DBTools Console the “HELP” command will launched by default and the available commands will be displayed on the main panel.





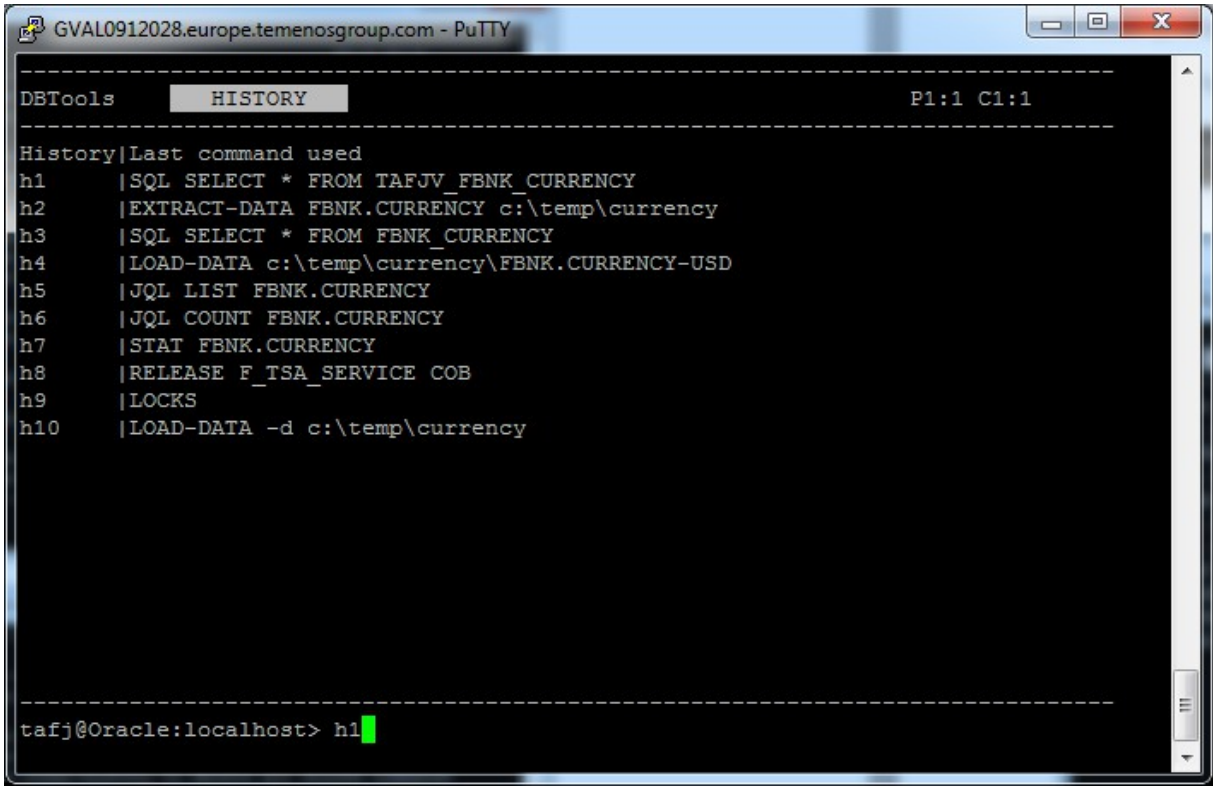
You can access this screen whenever you want by typing command “help” on the command line.

History

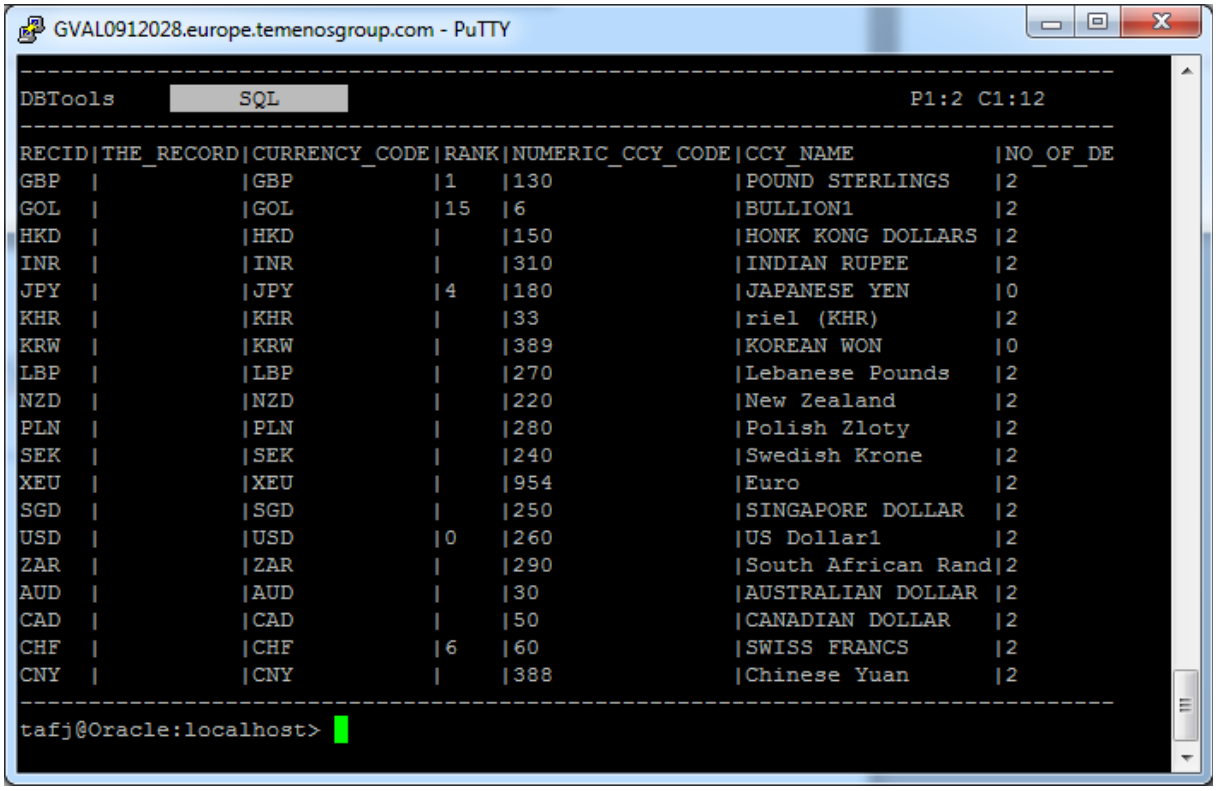
A history of last commands launched (10 last commands by default) is maintained to give the ability to reuse quickly last used command.

Simply enter “history” on the command line.

You will get the resulting output that shows the list of last used command.



Whether you are under history mode or not you can recall the command you want by typing its history index on the command line. i.e. you want to recall last SQL command just type h1 and you will get that command executed.



Please note that history is saved at the end of your session so you have the ability to load it the next time you will use DBTools Console. Data are stored in a properties file "DBTools" under TAFJ_HOME/conf.

If you don't want to keep that history for your next session, or to avoid other user to get it, as the DBTools properties file is shared between all users, you can clear it by typing command "clear-history".

As mentioned before DBTools Console doesn't rely on a user context, in consequence if many users use it at the same time the last one who quit his session will override the properties file with his settings.

Aliases

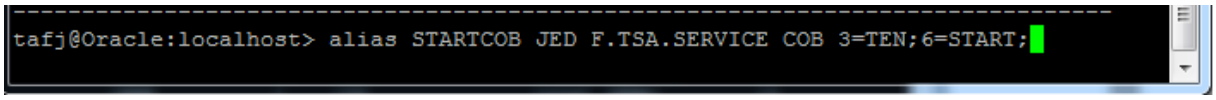
Create an alias

DBTools Console also provides an "alias" functionality to reuse complex command efficiently.

For example you often use the same JED statement to setup F.TSA.SERVICE, record COB, status to "START" and number of AGENT to "TEN" : JED F.TSA.SERVICE COB 3=TEN;6=START;

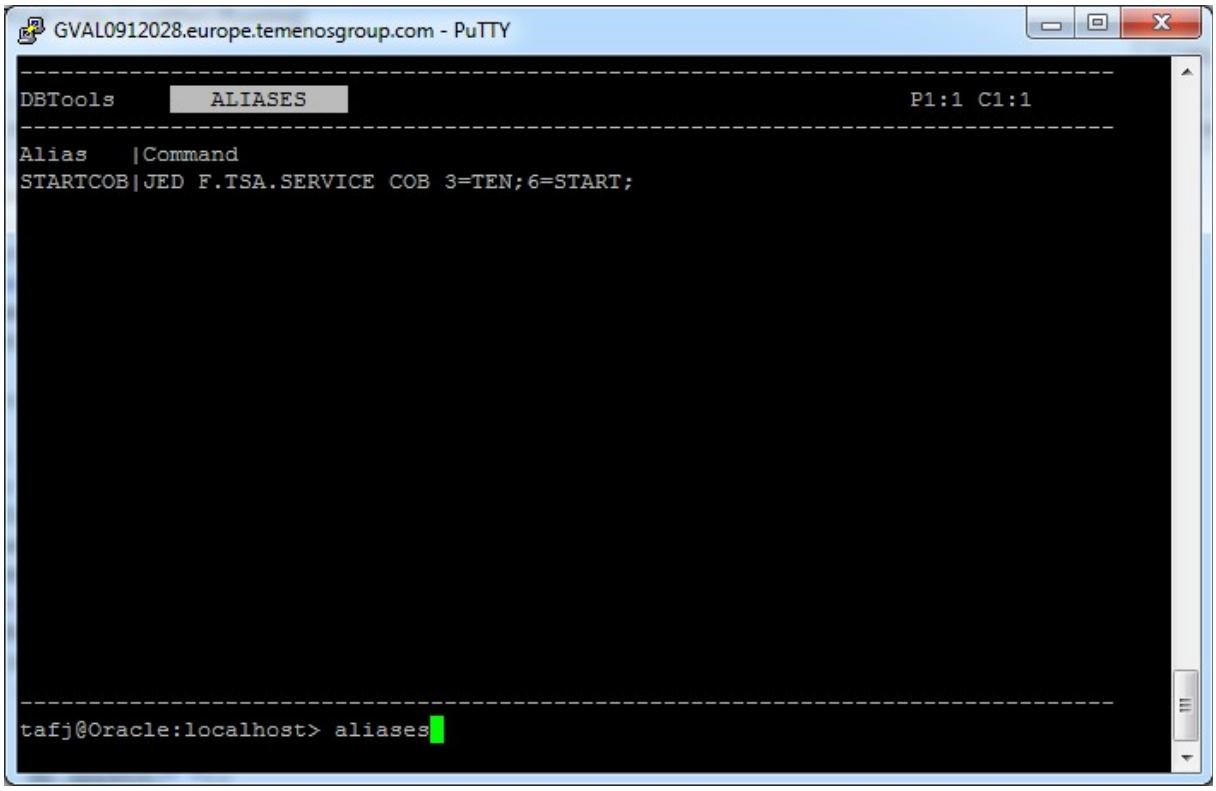
You can create an alias to create a shortcut to this command by typing :

alias <<ALIAS NAME>><<BLANK>><<COMMAND>>



Display aliases

To display all stored aliases you can type command “aliases”



Execute an alias

To execute this command just type the alias name on the command line.

Please note that if you create an alias for a SQL or JQL statement you have to switch to the appropriate mode before calling the alias or the command won’t be recognized.

i.e.

To execute this SQL alias switch to SQL mode first.

```
tafj@Oracle:localhost> SQL
```

Then you can execute alias.

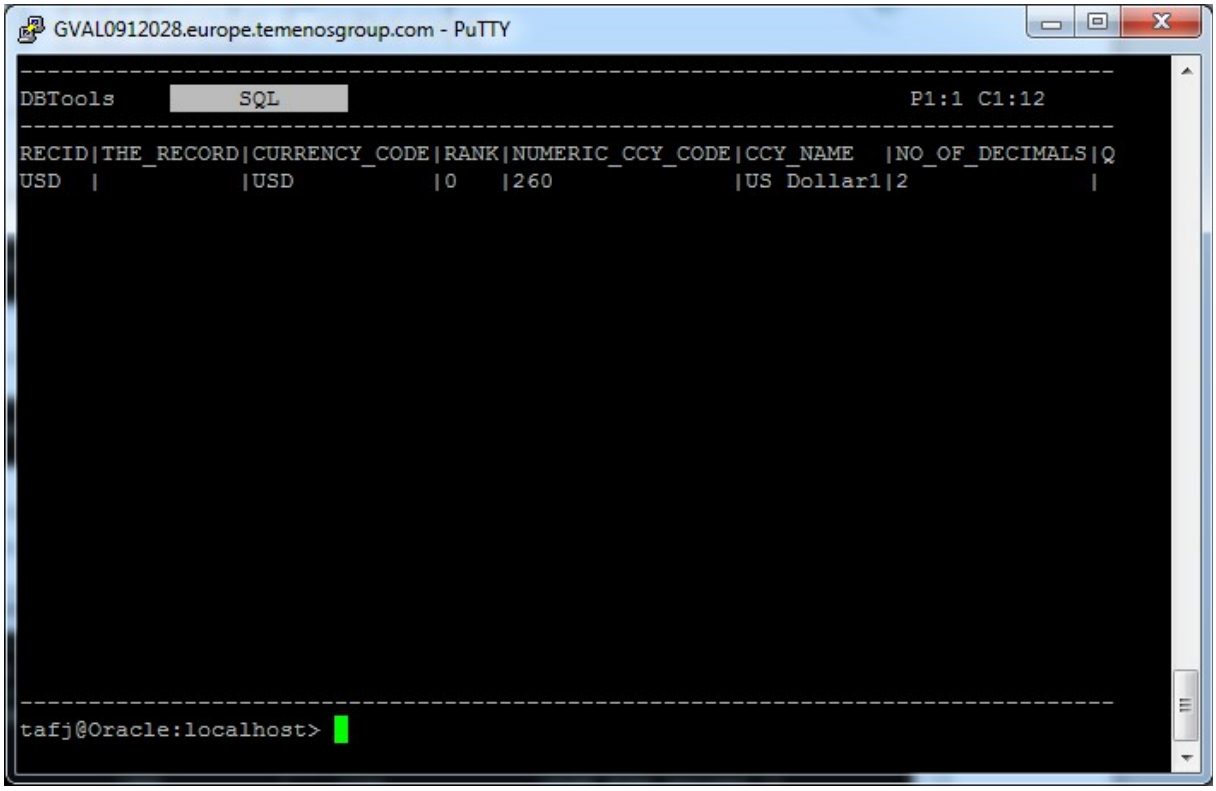
```
tafj@Oracle:localhost> CURRENCY
```

You have also the ability to complete an alias with an additional statement, i.e. you want to add criteria to a SQL statement.

GVAL0912028.europe.temenosgroup.com - PuTTY

```
DBTools        SQL                                              P1:2 C1:12
--------------------------------------------------------------------------------
RECID|THE_RECORD|CURRENCY_CODE|RANK|NUMERIC_CCY_CODE|CCY_NAME           |NO_OF_DE
GBP  |          |GBP          |1   |130             |POUND STERLINGS   |2
GOL  |          |GOL          |15  |6               |BULLION1          |2
HKD  |          |HKD          |    |150             |HONK KONG DOLLARS |2
INR  |          |INR          |    |310             |INDIAN RUPEE      |2
JPY  |          |JPY          |4   |180             |JAPANESE YEN      |0
KHR  |          |KHR          |    |33              |riel (KHR)        |2
KRW  |          |KRW          |    |389             |KOREAN WON        |0
LBP  |          |LBP          |    |270             |Lebanese Pounds   |2
NZD  |          |NZD          |    |220             |New Zealand       |2
PLN  |          |PLN          |    |280             |Polish Zloty      |2
SEK  |          |SEK          |    |240             |Swedish Krone     |2
XEU  |          |XEU          |    |954             |Euro              |2
SGD  |          |SGD          |    |250             |SINGAPORE DOLLAR  |2
USD  |          |USD          |0   |260             |US Dollar1        |2
ZAR  |          |ZAR          |    |290             |South African Rand|2
AUD  |          |AUD          |    |30              |AUSTRALIAN DOLLAR |2
CAD  |          |CAD          |    |50              |CANADIAN DOLLAR   |2
CHF  |          |CHF          |6   |60              |SWISS FRANCS      |2
CNY  |          |CNY          |    |388             |Chinese Yuan      |2
--------------------------------------------------------------------------------
tafj@Oracle:localhost> CURRENCY WHERE RECID='USD'
```

The resulting output will be.



Remove an alias

To delete an existing alias just type command unalias <<ALIAS NAME>>

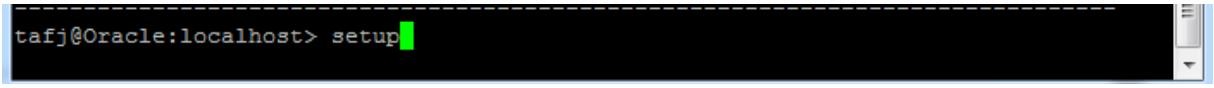
Please note that aliases are saved at the end of your session so you have the ability to load it the next time you will use DBTools Console. Data are stored in a properties file "DBTools" under TAFJ_HOME/conf.

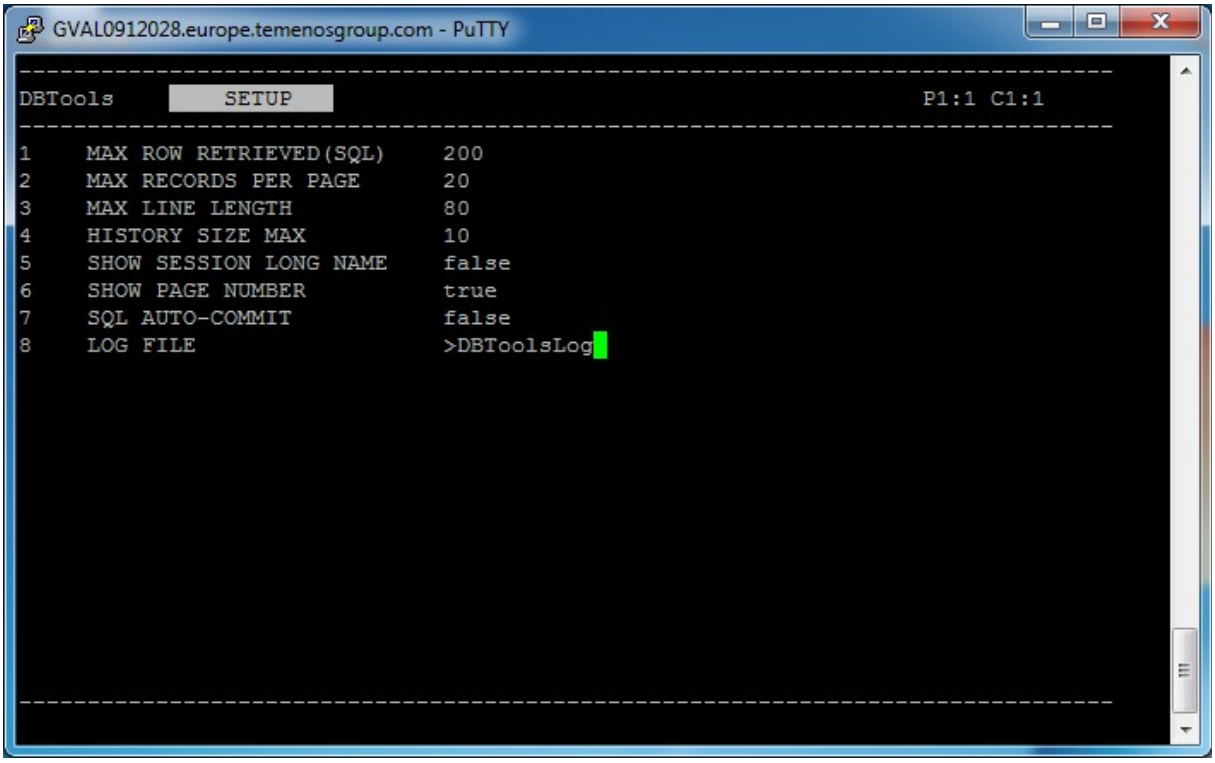
As mentioned before DBTools Console doesn't rely on a user context, in consequence if many users use it at the same time the last one who quit his session will override the properties file with his settings.

Setup

By typing command setup you will access to some properties allowing you to customize your session.

As mentioned before these properties are stored in the DBTools properties file and are shared between all users.





Max row retrieved (SQL)

This property allows you to customize the maximal number of rows that would be retrieved from a SQL SELECT statement.

It prohibits memory exception in case of large result set.

To be considered, value should be an Integer, default value is 200.

Max records per page

This property allows you to customize the maximal number of results that would be displayed in the main panel.

To be considered, value should be an Integer, default value is 20. Make sure your console is correctly sized to display the requested number of records.

Max line length

This property allows you to customize the maximal line length that would be displayed in the main panel.

To be considered, value should be an Integer, default value is 80. Make sure your console is correctly sized to display the full line length.

History size max

This property allows you to customize the maximal number of commands that would be stored in the history.

To be considered, value should be an Integer, default value is 10.

Show session long name

This property allows you to customize the session name that would be displayed in the footer panel. When set to ‘true’ session long name is applied.

To be considered, value should be a boolean, default value is false.

Show page number

This property allows you to display or not information about page number and column number in header panel.

To be considered, value should be a boolean, default value is true.

SQL auto-commit

This property allows you to enable / disable auto-commit on your database connection. When running SQL statements with auto-commit set to false, your change won’t be applied unless you execute a COMMIT statement before exiting.

To be considered, value should be a boolean, default value is false.

Log file

This property allows logging command result to the related log file by providing a log file name. Default value is null, which means command result won’t be logged.

Navigation and display

Change page/ column

For results that can't be displayed on a single page and/or a single column (by default 20 lines are being displayed per page and a line is 80 characters long), you have the ability to navigate between pages by using the following commands:

- **f** to go one page forward.
- **b** to go one page backward
- **sr** to scroll right to next column
- **sl** to scroll left to previous column

Change console display

When displaying a result you have the ability to fix the first line and / or the first column of the data being displayed for all the pages and columns.

It's especially useful in case of SQL SELECT where you might want to remain fixed the first line that displays column names, and the first column that displays the record id.

By default DBTools console will split result into columns in the case that the record to be displayed is too large to fit on one unique column. You could disable that functionality to have your record displayed on a single column.

Type the following commands:

- **hc** to enable / disable first fixed column
- **hl** to enable / disable first fixed line
- **cm** to enable / disable column mode

JED Console application

Syntax

TAFJ-JED application is designed to edit and modify database records by providing their file name and record key.

The JED Console application is made to replace JED GUI application on platforms with possibly no X Window installed. : UNIX, AIX, z/OS.

JED Console should also be faster than the current SWING based application, and built upon TAFJ-Core framework to be independent of any native libraries.

It could be called from DBTools console and in standalone mode too.

Execution – Interactive mode

JED Console could be launched from TAFJ_HOME/bin.

```
tJed [-cf <configuration>[.properties]] <file name> <record key>
```

The targeted database will be the one defined in the default configuration file or in the configuration specified by the -cf option.

Example
To edit the record SYSTEM from SPF file.

```
tJed F.SPF SYSTEM
```

Optional parameters

Fields per page number
By default JED console is configured to display 20 fields per page.

By providing option line: **-l <fields per page number>** you will be able to change dynamically this property. Please ensure that your terminal is correctly sized to display the number of line required or the data won't be displayed properly.

```
tJed -l 50 F.SPF SYSTEM
```

Column names

By default JED console is configured to display a record with no field name.

If connected to T24 and by providing the column name option: **-cn** you will be able to retrieve the record field names if they are defined in the related T24 application.

```
tJed -cn F.SPF SYSTEM
```

Help

Before editing your record, you have the ability to display JED console available functionalities by providing the help option: **-help**

```
tJed -help F.SPF SYSTEM
```

LOCK

By default JED console will read the requested record with no lock on it. To make a ***readu*** you have to provide the **LOCK** parameter with value **Y**.

```
tJed  F.SPF SYSTEM LOCK=Y
```

JED Graphical mode

By providing option graphical user interface: **-gui** you will be able to launch JED in graphical mode.

```
tJed  -gui F.SPF SYSTEM
```

Execution - script mode

It is also possible to modify a record by using JED console from command line, without opening it. That's what we call "script mode".

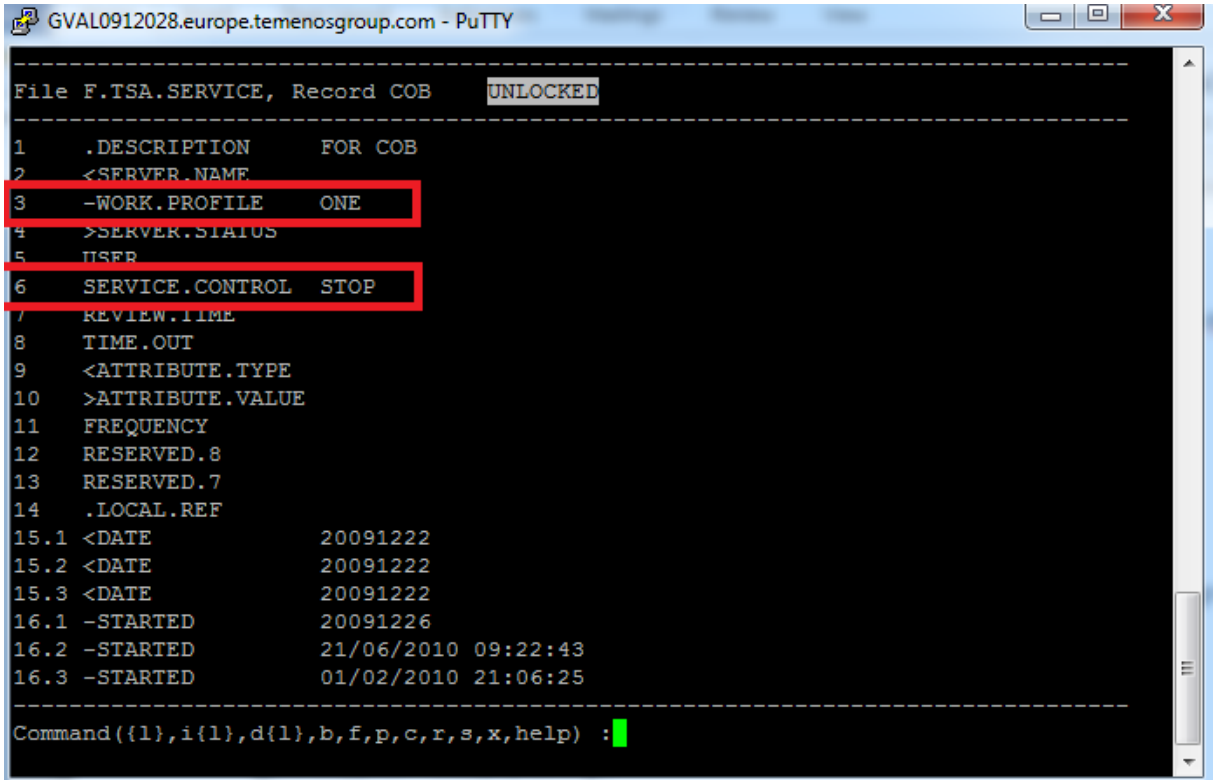
To update the record **you just have to add '-s' to the command and know the field(s) number you want to change** and provide the associated values.

Example

To change in file F.TSA.SERVICE, COB record, field value number of agent (WORK.PROFILE) and service status (SERVICE.CONTROL), you have to do the following:

If you don't have any idea of the related field numbers, firstly you have to open the record in console mode and get these numbers.

```
tJed  F.TSA.SERVICE COB
```



We can see that the field numbers we want to update are 3 and 6.

Then you can type the following command to change these fields to “TWO” and “START” :

```
tJed -s F.TSA.SERVICE COB 3=TWO;6=START;
```

Note: depending on the operating system you could have to escape the part of command after -s option with quotes: JED -s ‘F.TSA.SERVICE COB 3=TWO;6=START;’

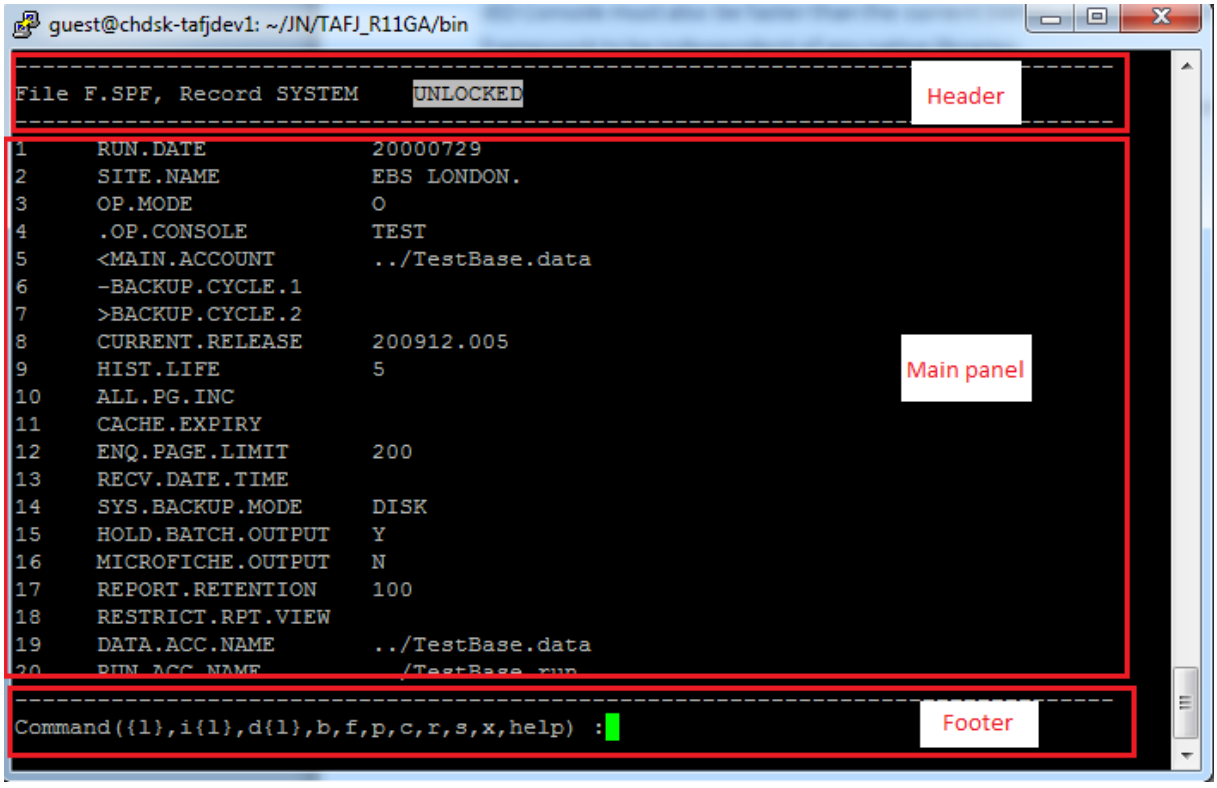
The record will be automatically updated to its new value without opening the console.

It also works for multi-value and sub-value. i.e. tJed -s MY.FILE MYRECORD 1.1=VM;2.1.1=SM;

Presentation

JED console consists of three panels:

- Header panel that contains information about the record
- Main panel that contains the record to be edited
- Footer panel that contains the command line



Header

The header presents information related to the edited record:

- File name
- Record key

- Record lock status : the value could be UNLOCKED (default) / LOCKED depending on LOCK parameter provided

```
File F.SPF, Record SYSTEM     UNLOCKED
```

File name Record key Record lock status

```
File F.SPF, Record SYSTEM     LOCKED
```

- Record status: the value could be **empty** (default, that means no current change made to the record) / **WAITING** (when editing a field, waiting for user entry) / **PENDING** (change made to a record but not validated into database yet) / **SAVED** (change made to the record validated in database).

Record status

```
File F.SPF, Record SYSTEM     LOCKED     WAITING
1       >20000729<
```

```
File F.SPF, Record SYSTEM     LOCKED     PENDING
1       20000730
```

```
File F.SPF, Record SYSTEM     LOCKED     SAVED
```

- Console display mode : value could be empty (default, that means large record will be displayed on multiple line and on a single column) / **COLUMN MODE** (that means large record will be displayed on single line and multiple column)

```
File F.SPF, Record SYSTEM     UNLOCKED
1       20100107
2       SINGLE COMPANY TRADE DATED SINGLE COMPANY TRADE DATED SINGLE COMPANY TRADE
2        DATED SINGLE COMPANY TRADE DATED SINGLE COMPANY TRADE DATED SINGLE COMPAN
2       Y TRADE DATED
```

```
File F.SPF, Record SYSTEM     UNLOCKED              COLUMN MODE
1       20100107
2       SINGLE COMPANY TRADE DATED SINGLE COMPANY TRADE DATED SINGLE COMPANY TRADE
```

Main panel

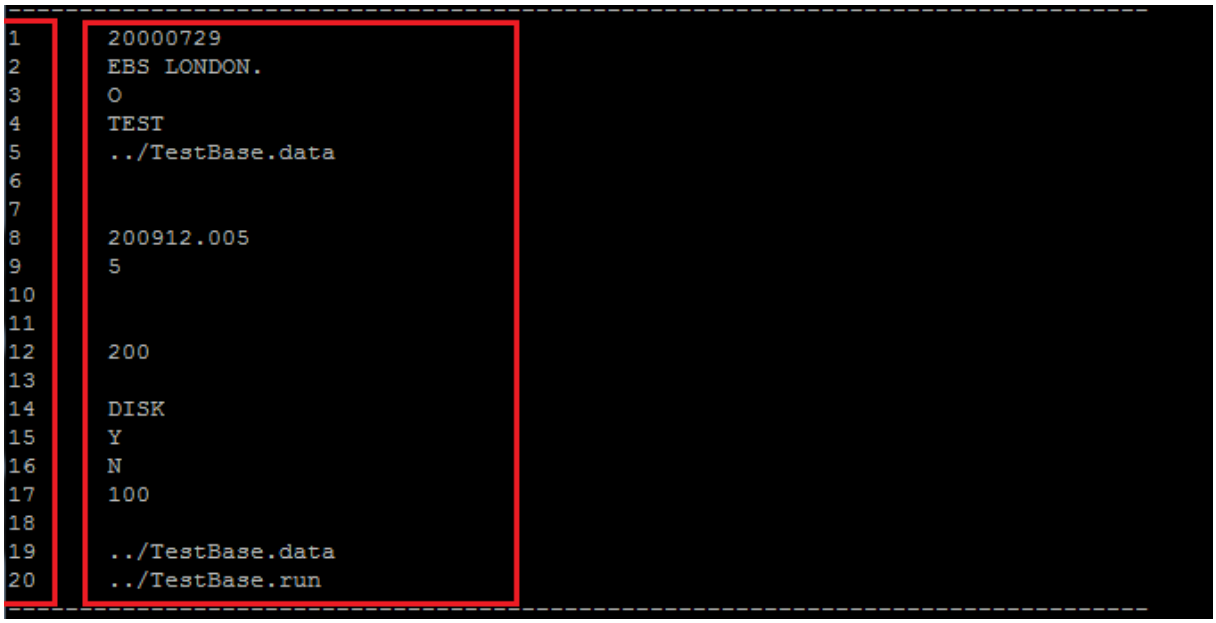
The main panel contains the record to be displayed. The record is displayed vertically and is split into lines. Each line has a number that correspond to a field, a multi-value (i.e. 1.1) or a sub-value (i.e. 1.1.1).

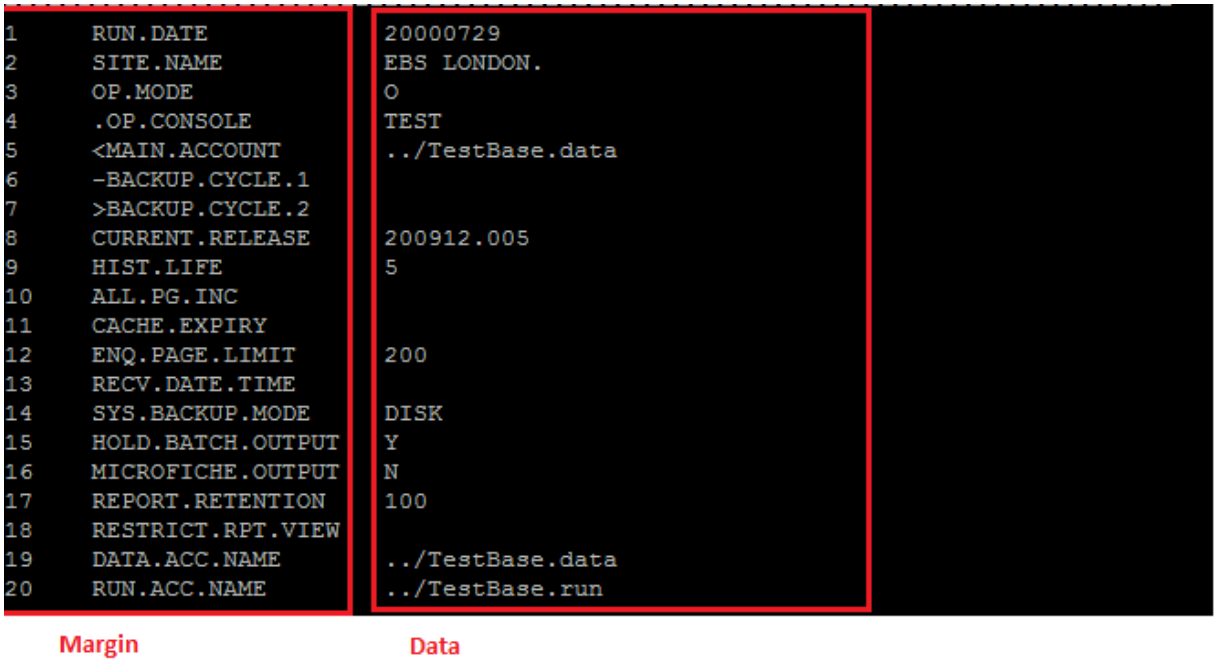
The field number is displayed on a margin on the left part of the panel; the associated value is displayed next to the margin on the middle of the panel.

JED console can display 80 characters per row, by default the margin is 5 characters long and the data part 75 characters long.

When connected to T24 it's possible to display in the margin the field names if they are defined, by providing **-cn** option.

In that case the margin length will increase and the data length will decrease.

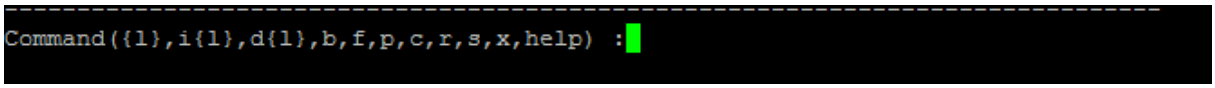




Footer

The footer panel contains the command line that shows some of the available commands. It can also contain some instructions related to console current mode.

Classic mode, waiting for user command



Editing mode, waiting for user validation

Functionalities

Change a record

Edit an existing field / Multi-value / Sub-value

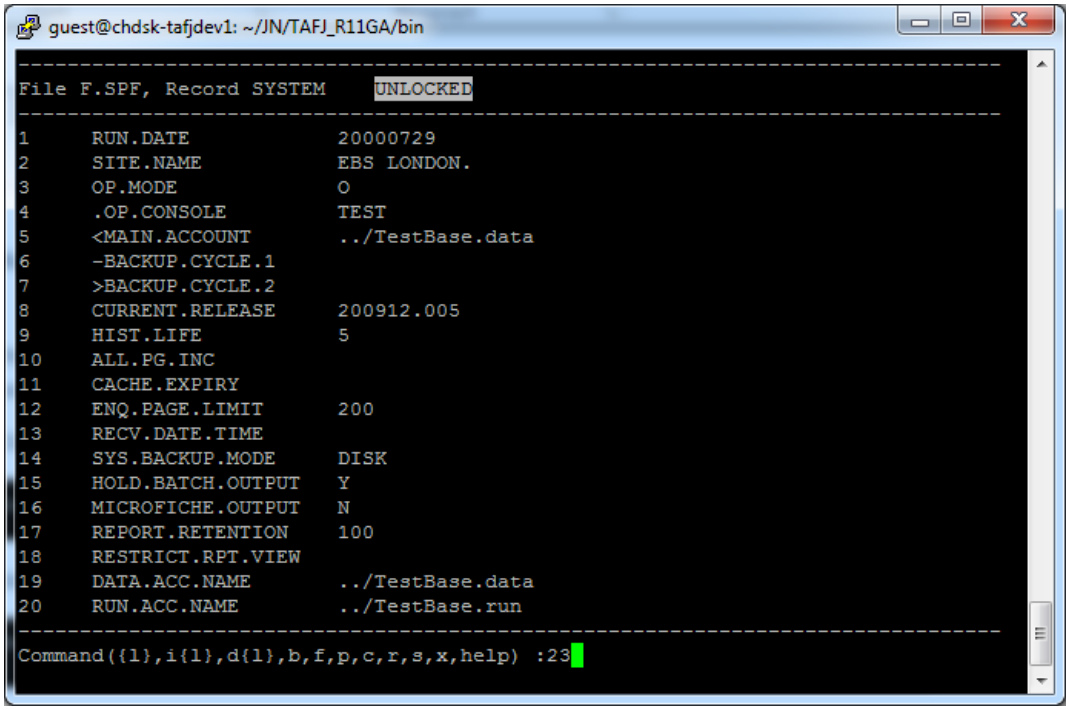
To edit an existing field, multi-value or sub-value just type the field, VM, SM number on the command line, even if the field isn't on the current page.

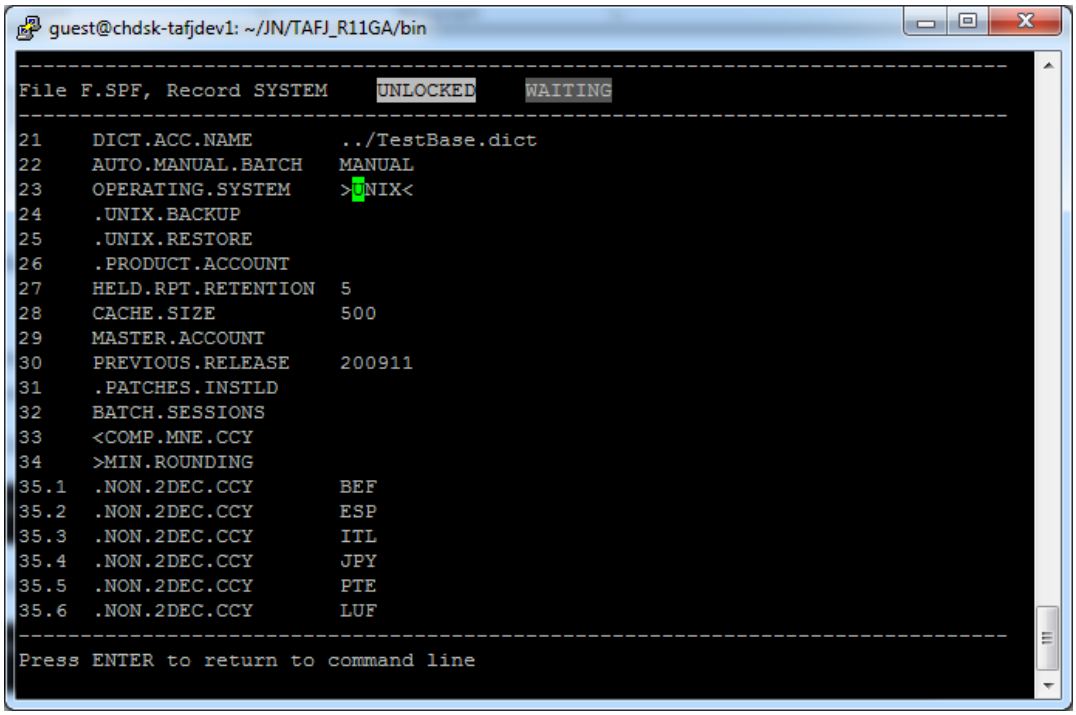
The prompt will be placed on the related page and line and you will be able to change the field value.

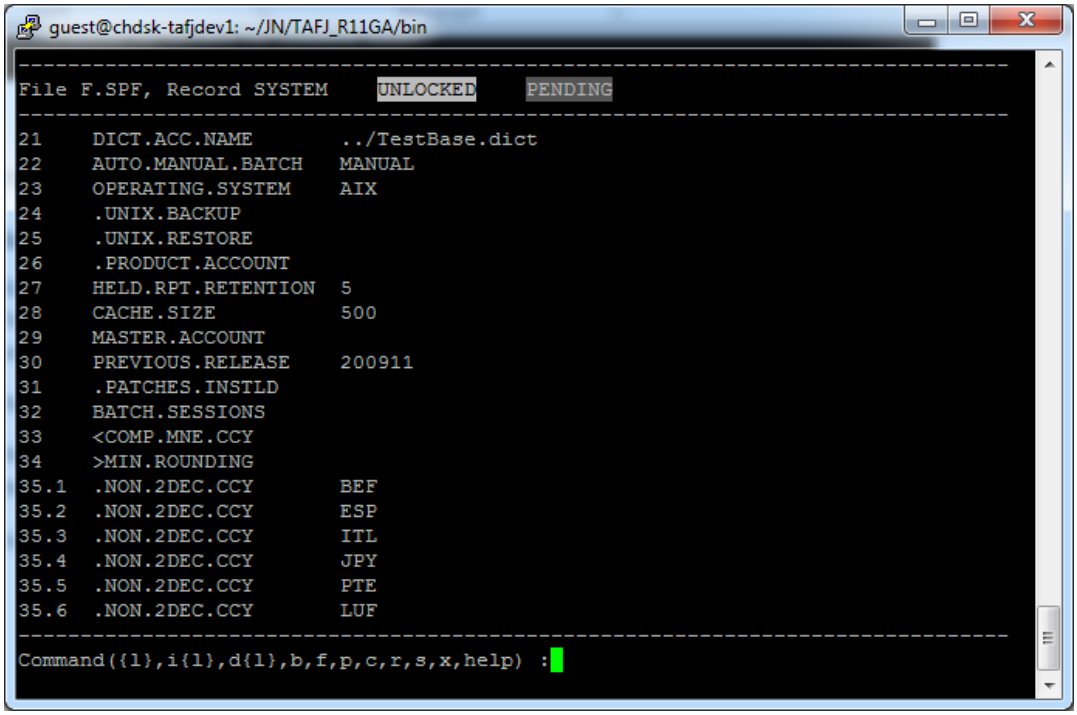
The current field value is placed into bracket that allows you to see if there are blank characters in the current value.

To validate the change, type “enter” and you will return to command line.

You will see the record status changing from WAITING to PENDING during this process.







Insert a field / Multi-value / Sub-value

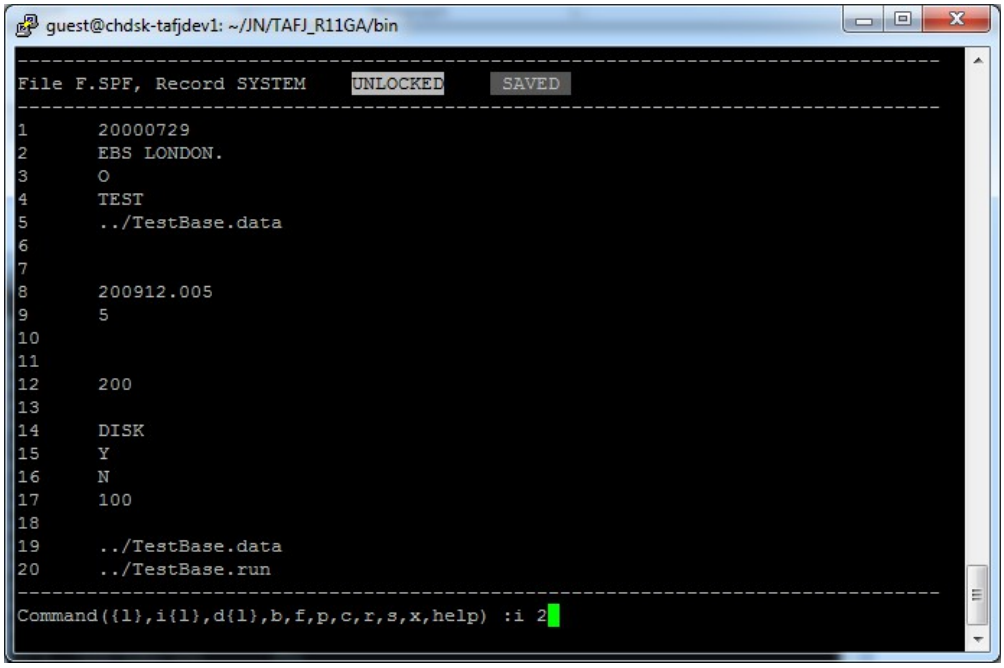
To insert a new field, multi-value or sub-value just type the command **i** followed by the field, VM, SM number on the command line.

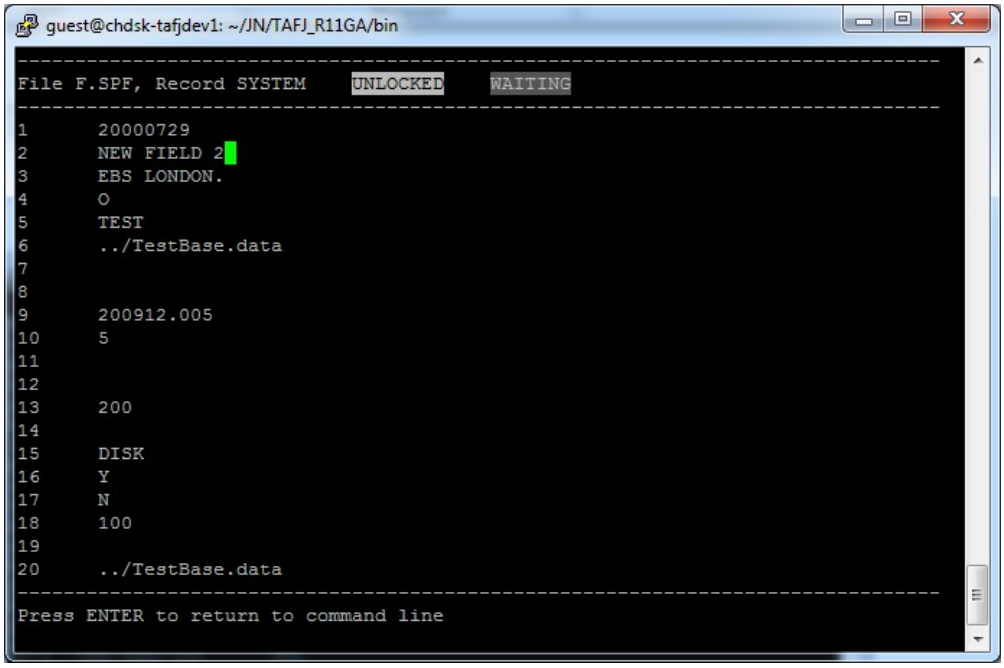
The prompt will be placed on the related page and line and you will be able to insert the field value.

To validate the insertion, type “enter” and you will return to command line.

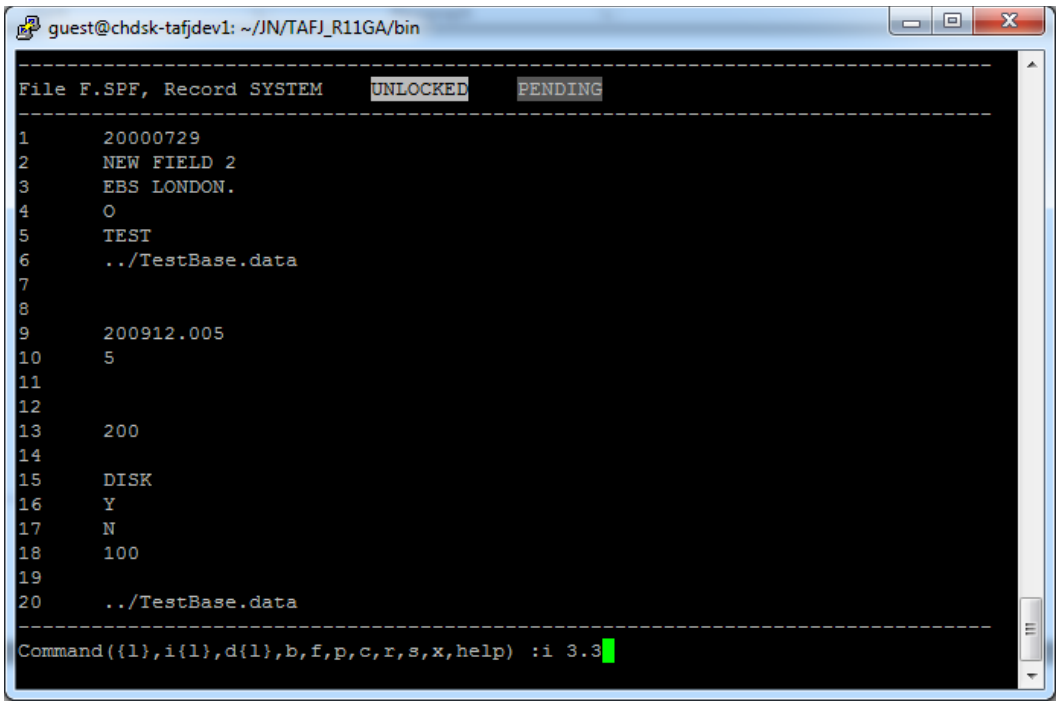
You will see the record status changing from WAITING to PENDING during this process.

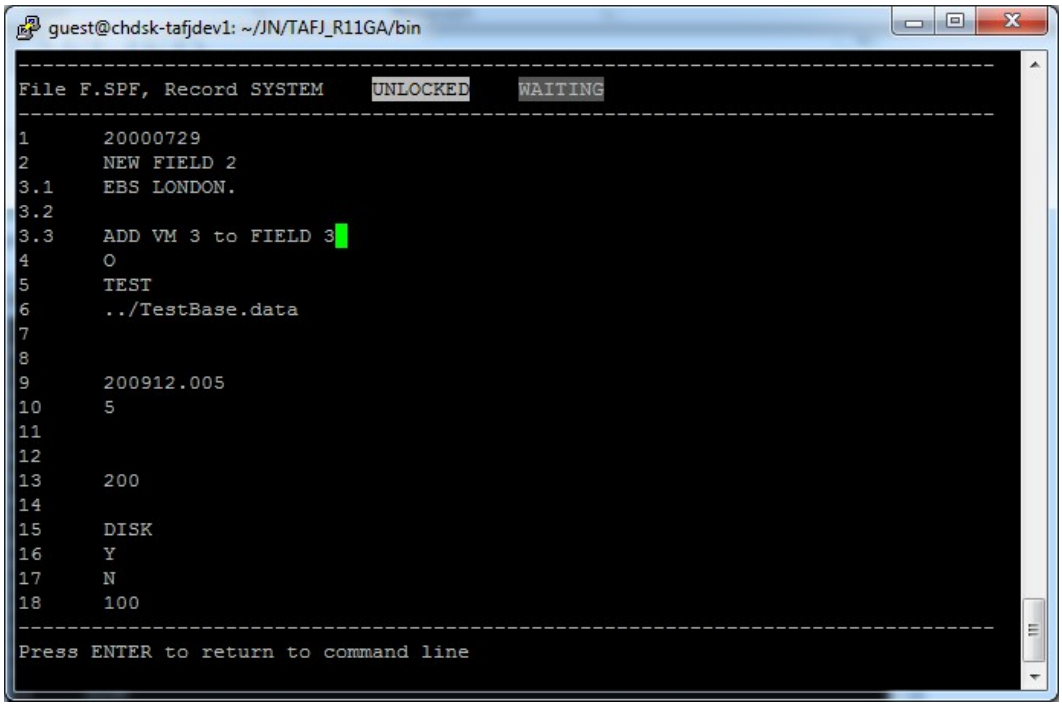
i.e. **i 2** to insert a field at the second position in the current record.



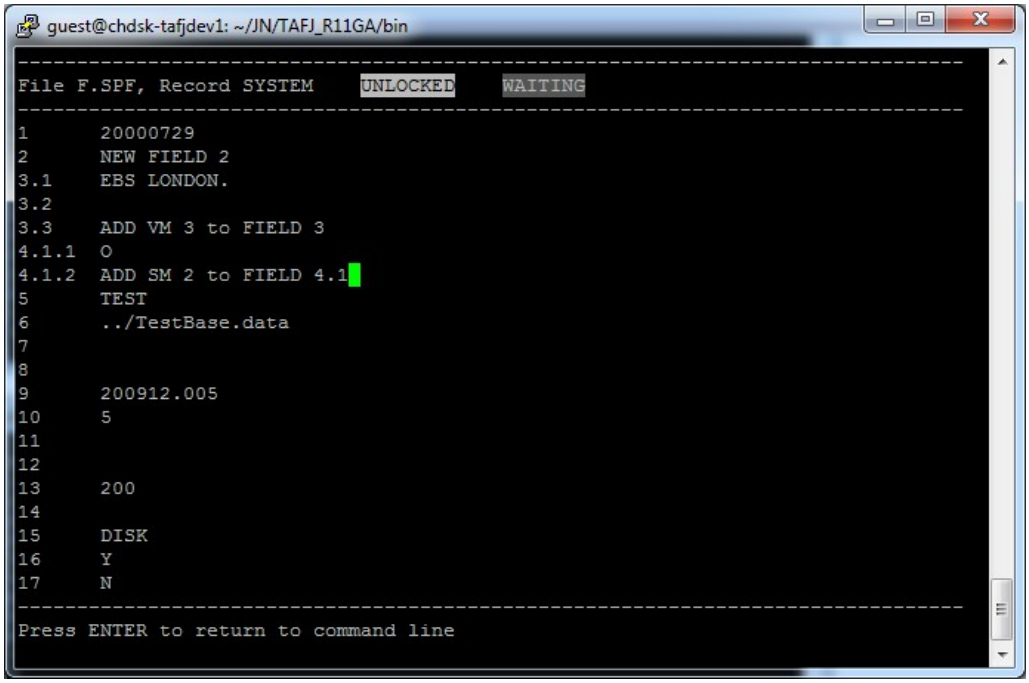


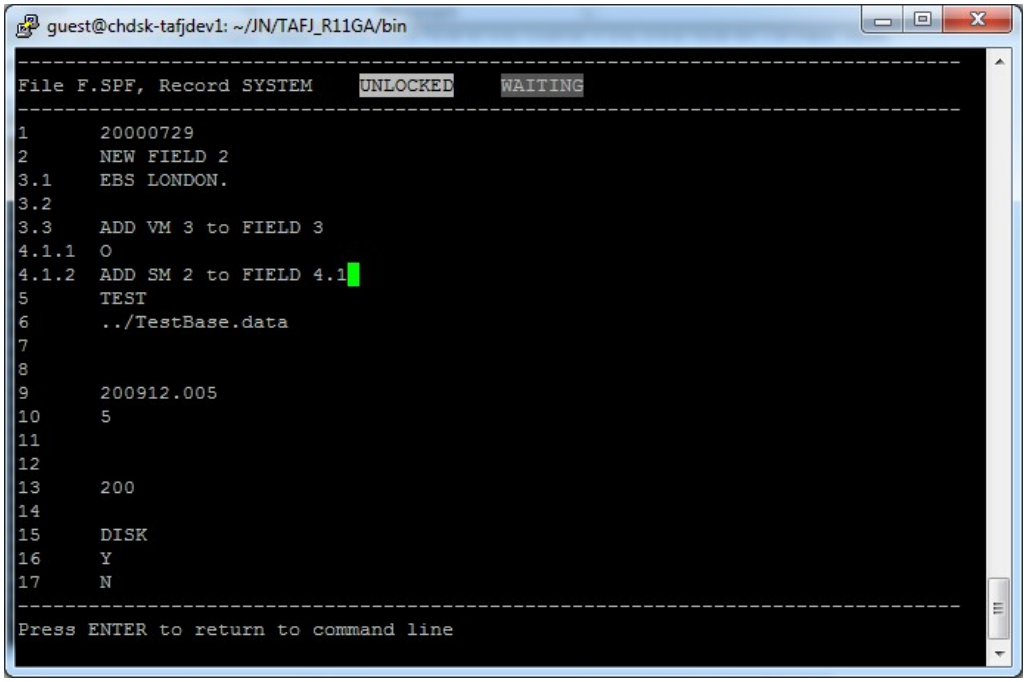
i.e. **i 3.3** to insert a third multi-value in the third field of the record. If the third field isn't a multi-value yet, its current value will be set as multi-value 1 and multi-value 2 and 3 will be added.





i.e. **i 4.1.2** to insert a second sub-value in the first multi-value of the fourth field of the record. If the fourth field isn't a multi-value neither a sub-value yet, its current value will be set as multi-value 1, sub-value 1 and sub-value 2 will be added.


guest@chdsk-tafjdev1: ~/JN/TAFJ_R11GA/bin
File F.SPF, Record SYSTEM UNLOCKED WAITING
1 20000729
2 NEW FIELD 2
3.1 EBS LONDON.
3.2
3.3 ADD VM 3 to FIELD 3
4.1.1 O
4.1.2 ADD SM 2 to FIELD 4.1
5 TEST
6 ../TestBase.data
7
8
9 200912.005
10 5
11
12
13 200
14
15 DISK
16 Y
17 N
Press ENTER to return to command line


guest@chdsk-tafjdev1: ~/JN/TAFJ_R11GA/bin
File F.SPF, Record SYSTEM UNLOCKED WAITING
1 20000729
2 NEW FIELD 2
3.1 EBS LONDON.
3.2
3.3 ADD VM 3 to FIELD 3
4.1.1 O
4.1.2 ADD SM 2 to FIELD 4.1
5 TEST
6 ../TestBase.data
7
8
9 200912.005
10 5
11
12
13 200
14
15 DISK
16 Y
17 N
Press ENTER to return to command line

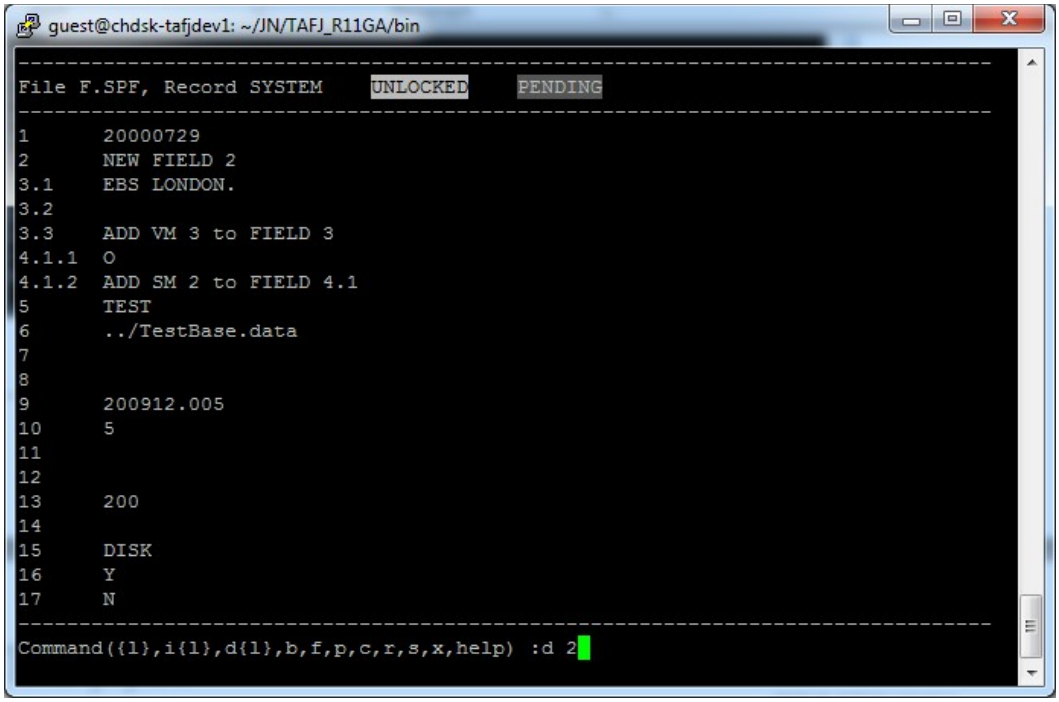
Delete a field / Multi-value / Sub-value

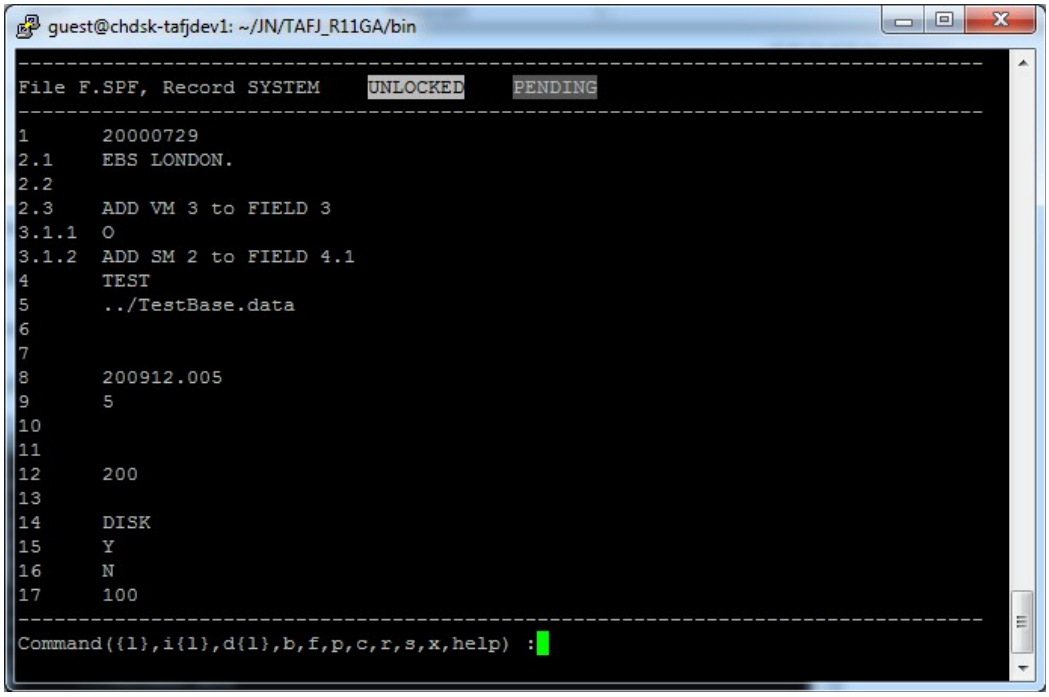
To delete a field, multi-value or sub-value just type the command **d** followed by the field, VM, SM number on the command line.

The fields, multi-values, sub-values that follow the deleted field will be renumbered.

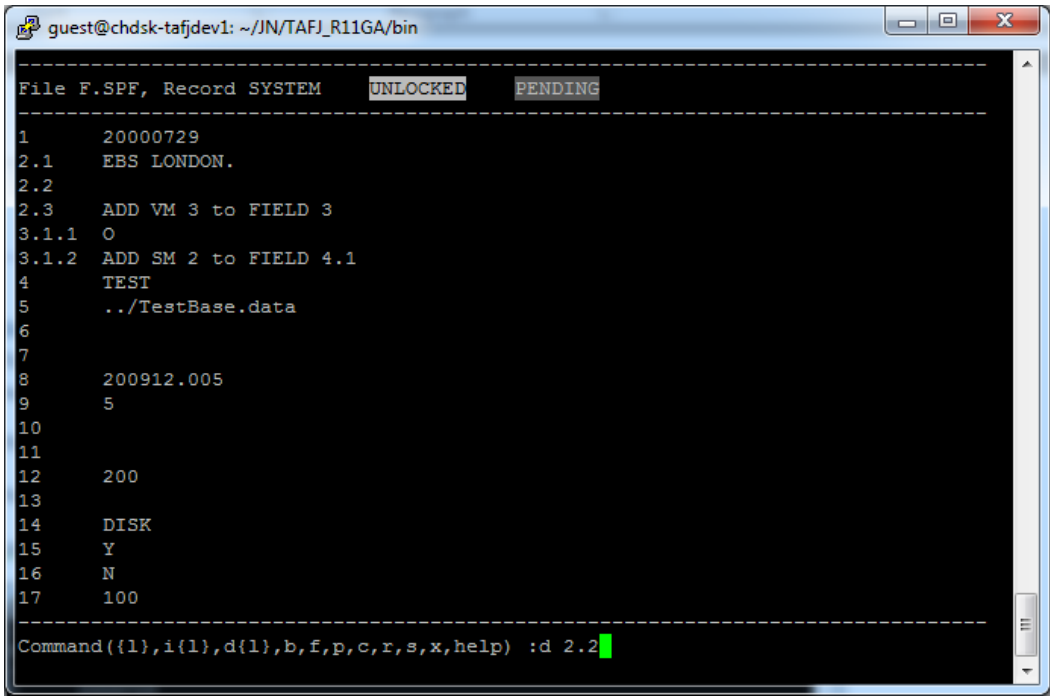
You will see the record status changing to PENDING during this process.

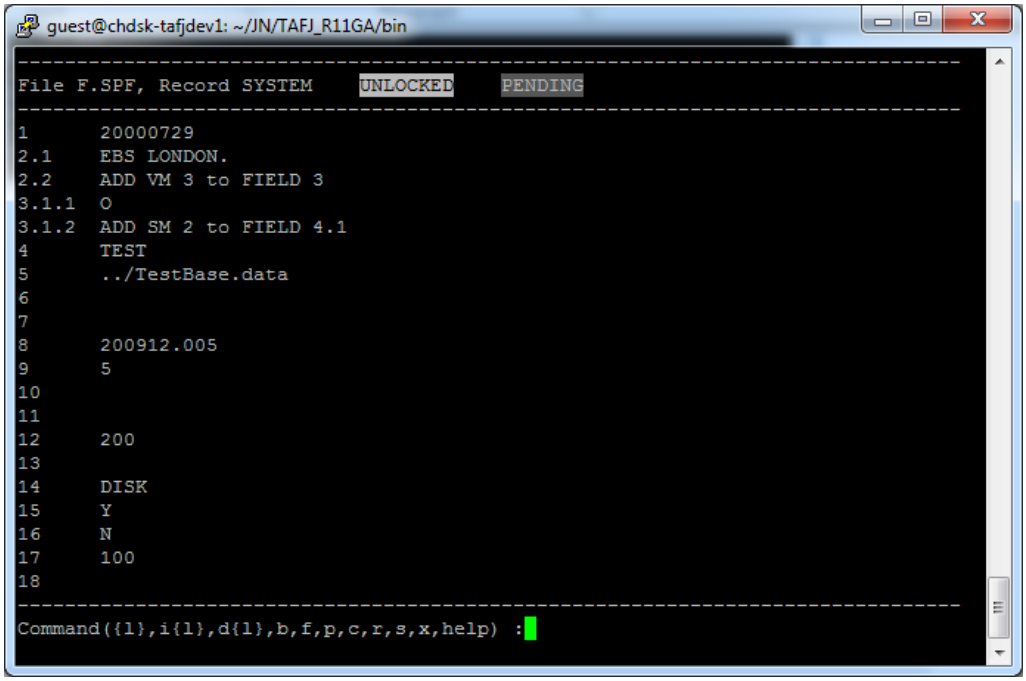
i.e. **d 2** to delete field number 2



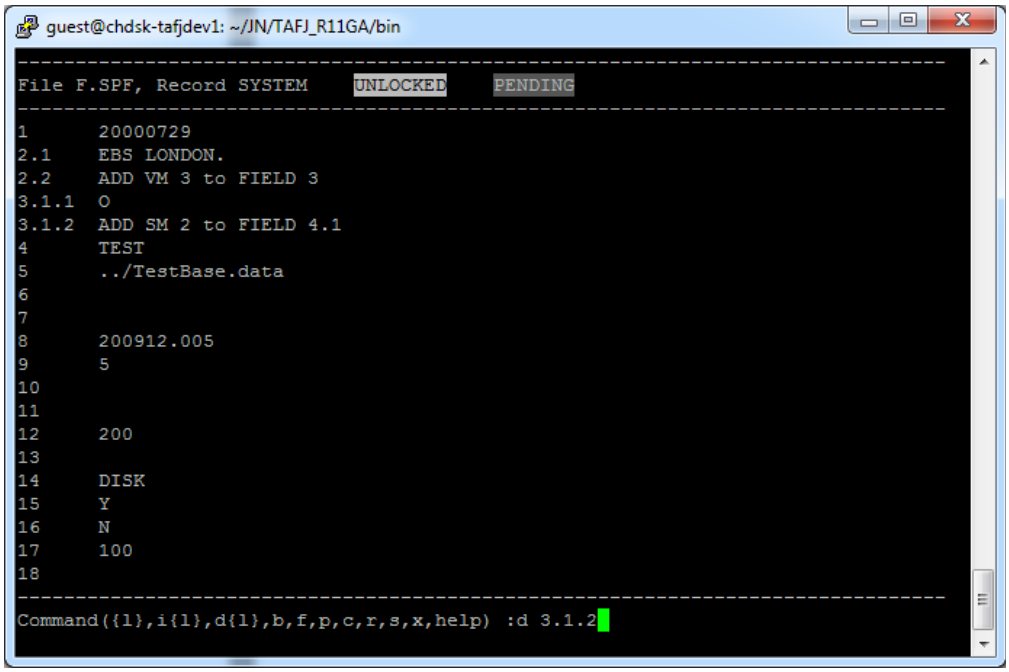


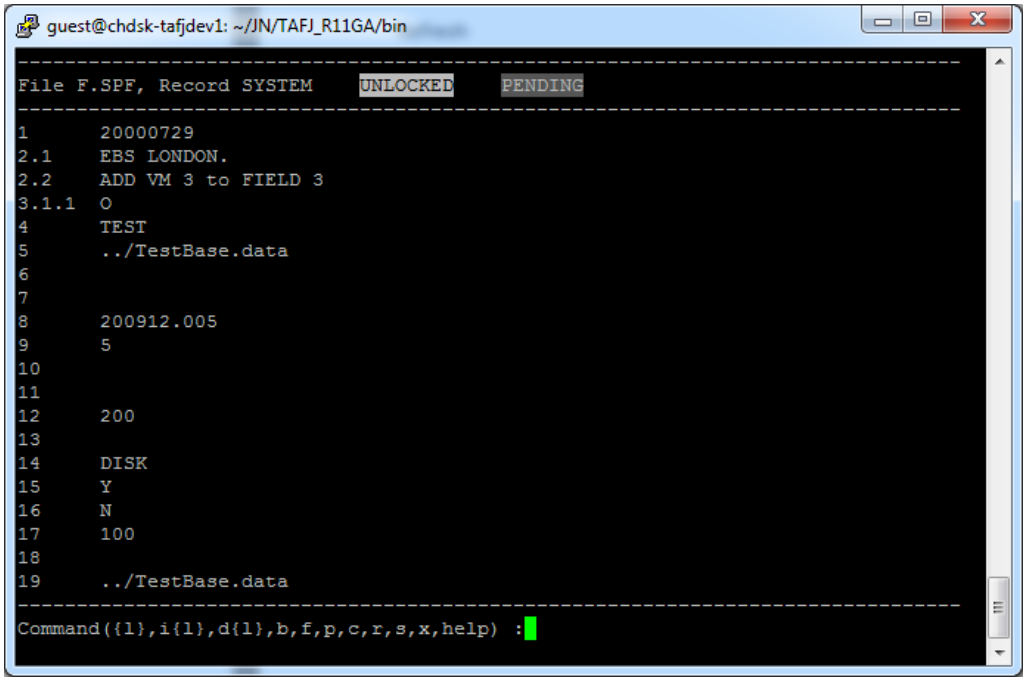
i.e. **d 2.2** to delete multi-value 2 of field 2





i.e. **d 3.1.2** to delete sub-value 2 of multi-value 1 of field 3

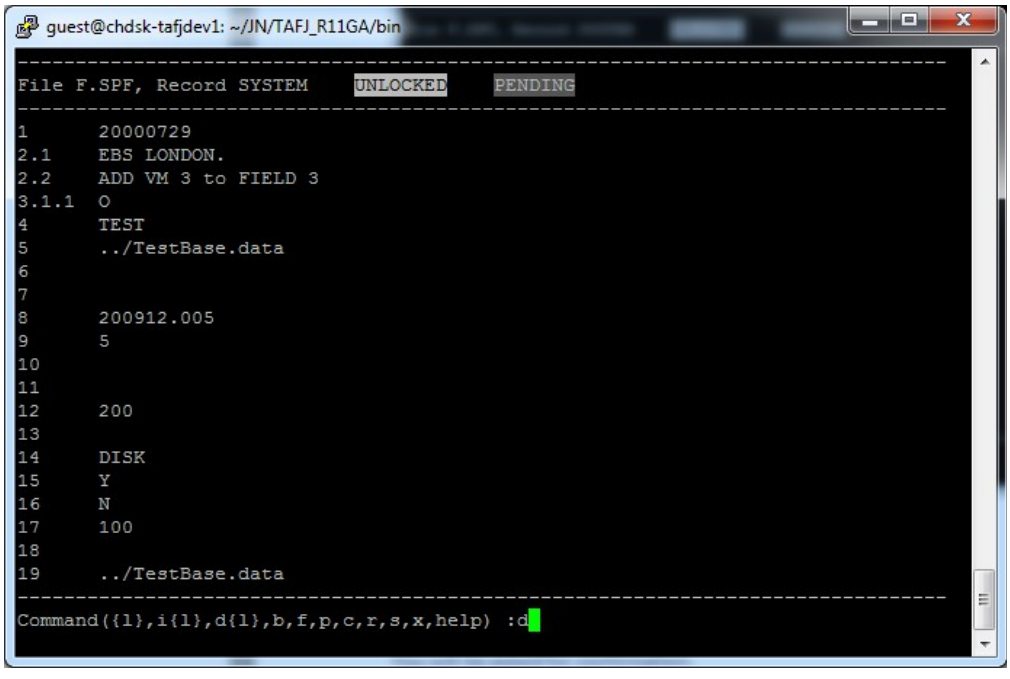


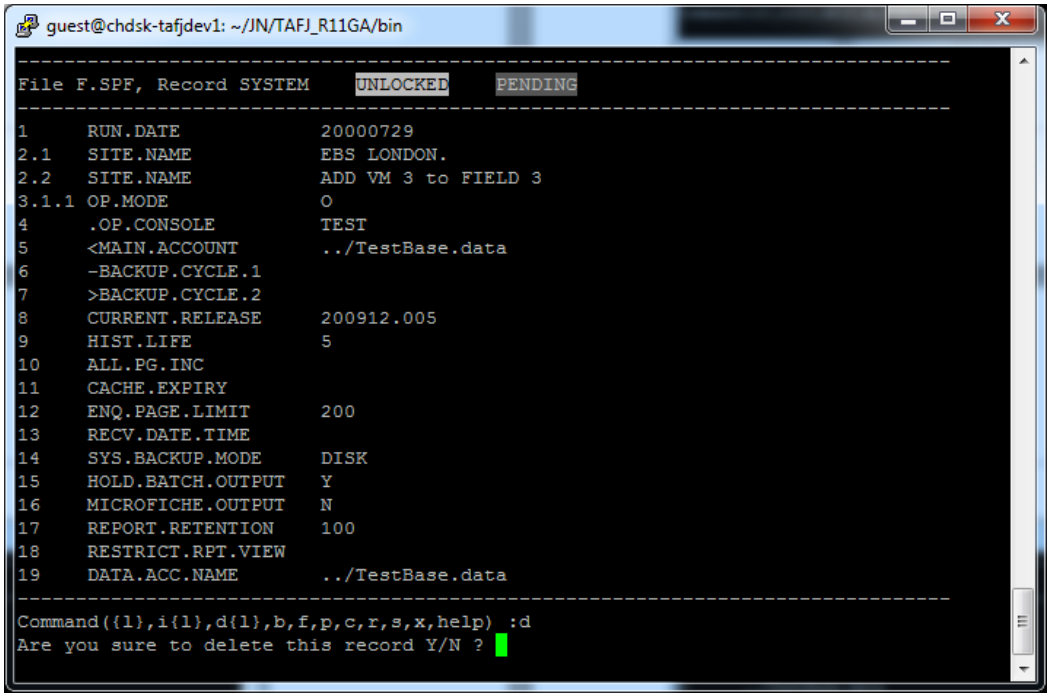


Delete a record

You can delete a record by typing command **d.**

You will be asked for confirmation.

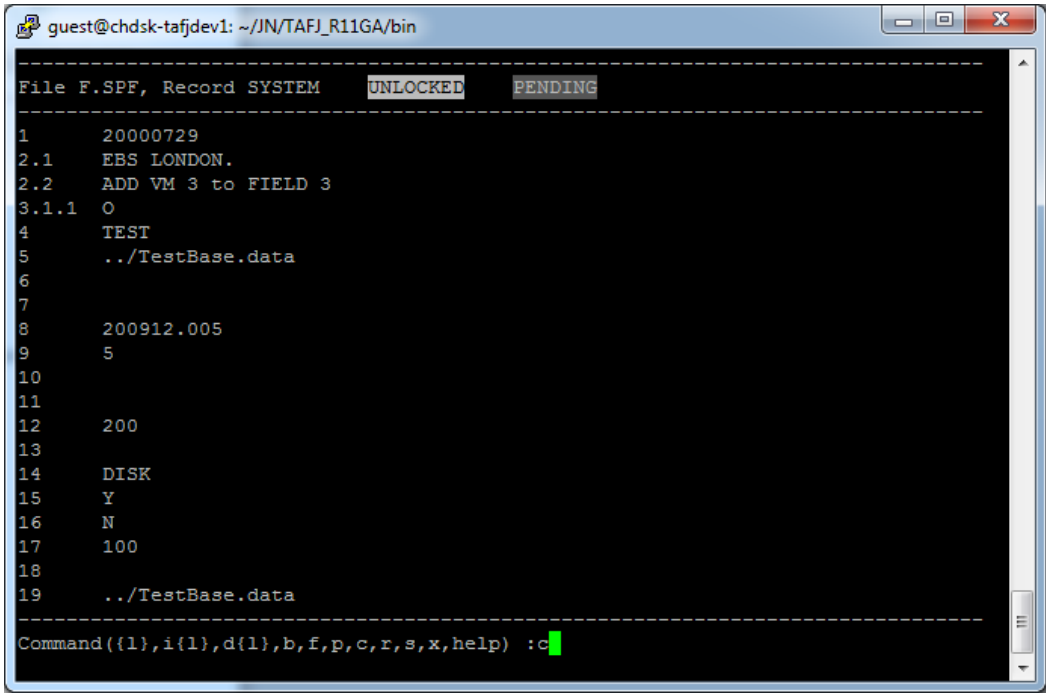


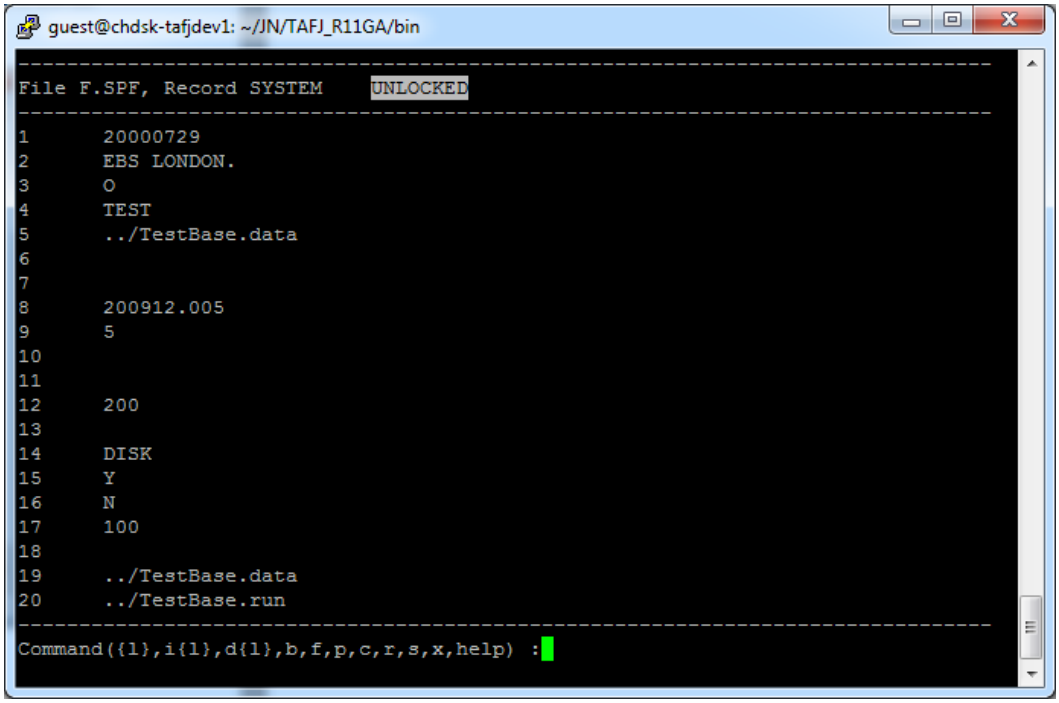


Validation

Cancel pending changes

You have the ability to cancel all your pending modifications by typing command **c.** The record will be restored at its initial state or at its state just after your last save. You will see the record status changing from PENDING to empty during this process.

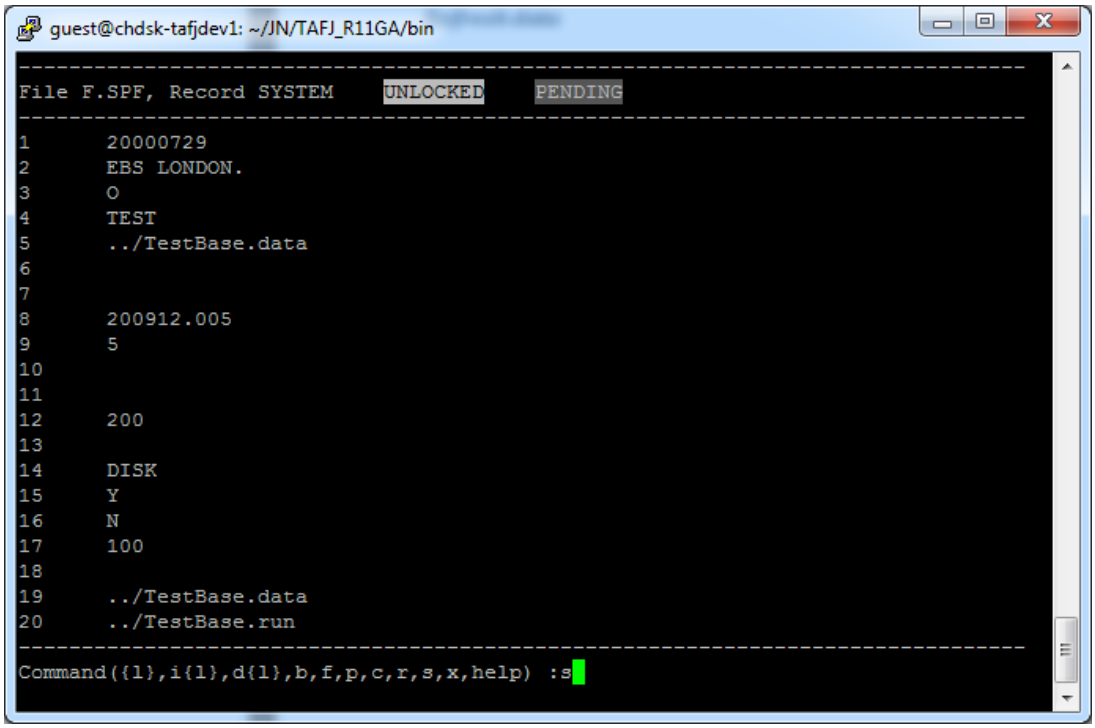


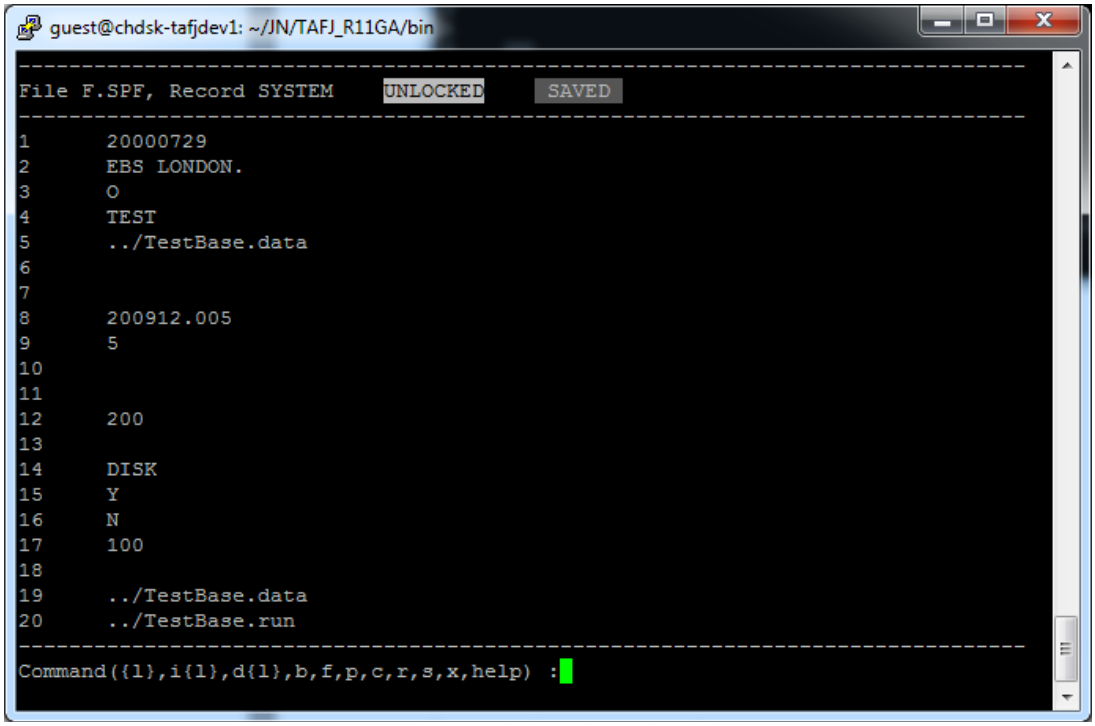


Save pending changes

To update your pending modifications into database, you can use command **s.**

You will see the record status changing from PENDING to SAVED.

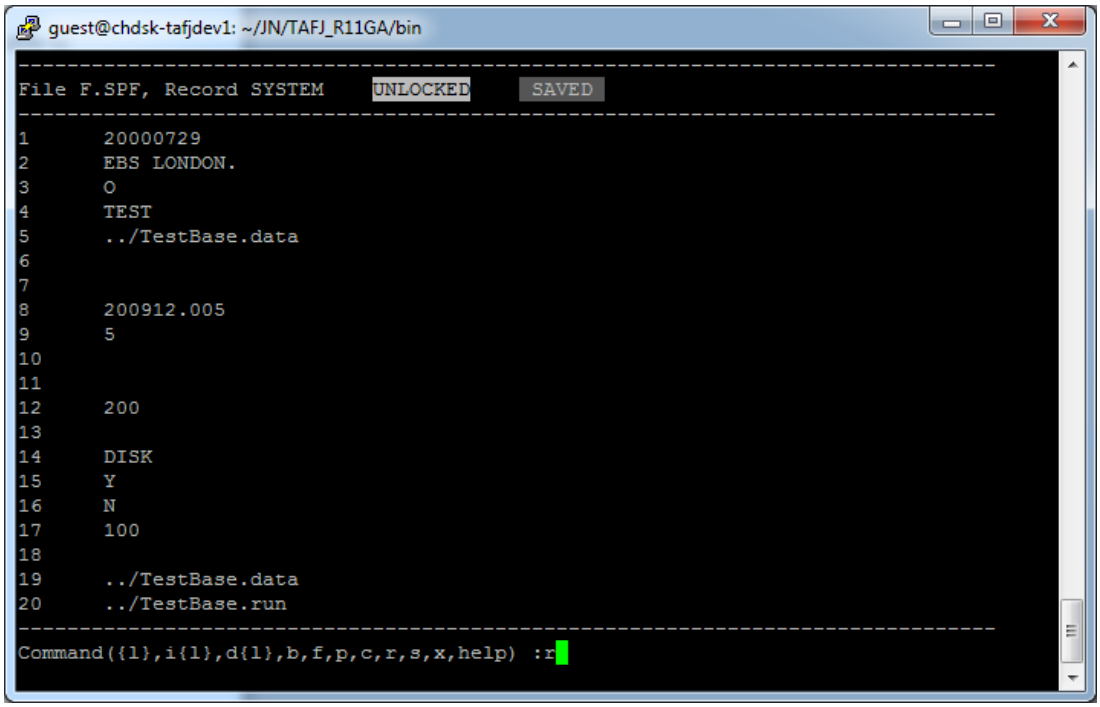


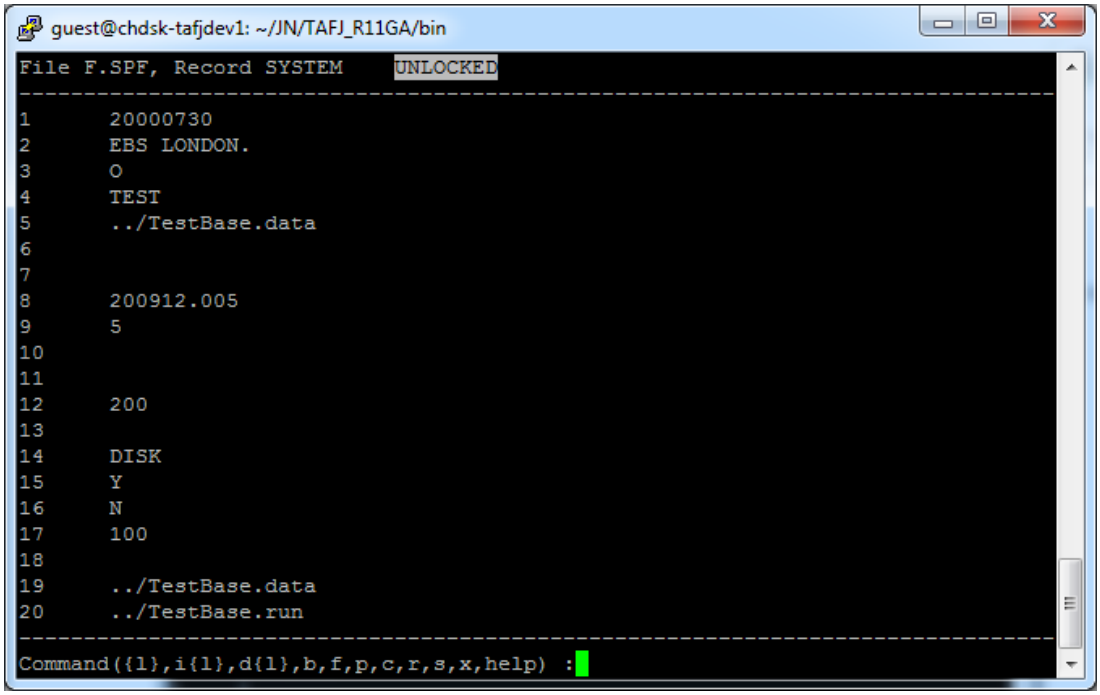


Refresh data

The command **r** will allow you to reload the record from the database, that way you will be able to see external modification that have been applied to the record since you have loaded it.

The record status will be set to its default empty value.





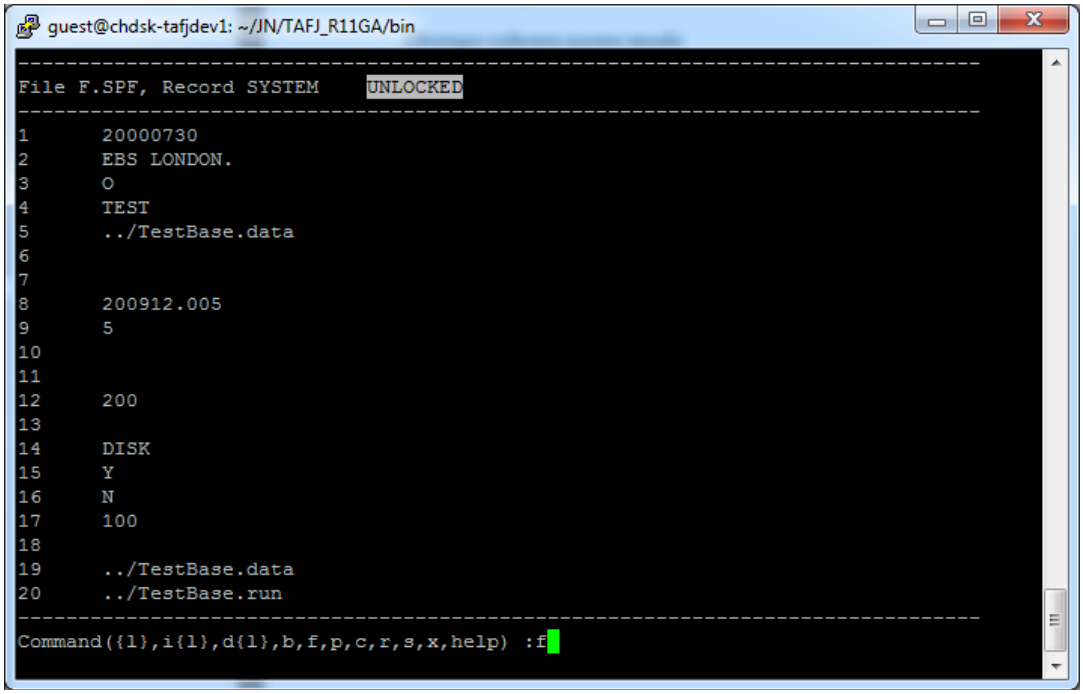
Navigation and display

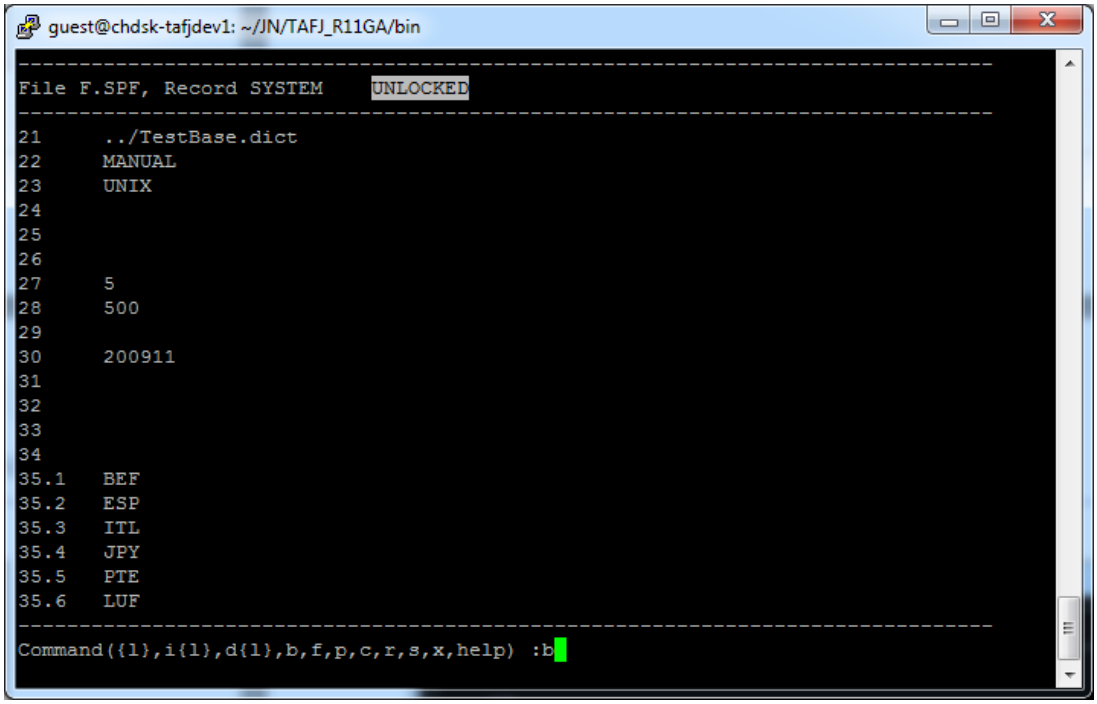
Change page

For record that can't be display on a single page (by default 20 lines are being displayed per page), you have the ability to navigate between pages by using the following commands:

- **f :** will make you move to the next page forward
- **b :** will make you go move previous page backward

i.e. **f** to move one page forward





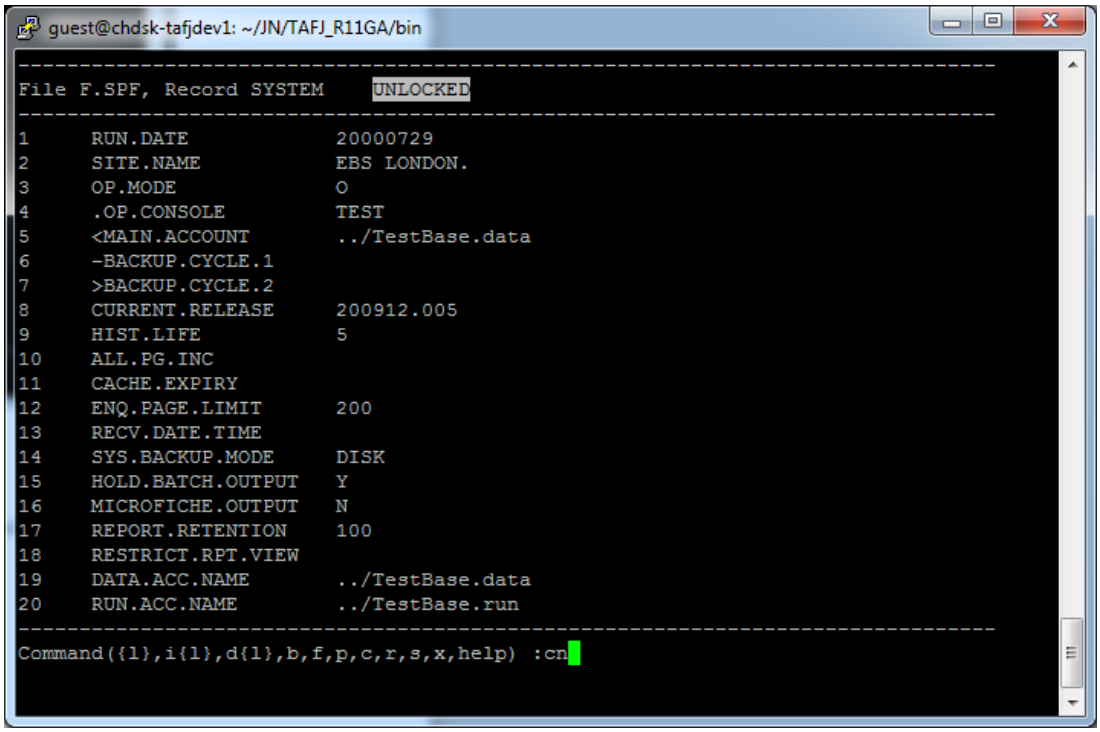
i.e. **b** to move to previous page

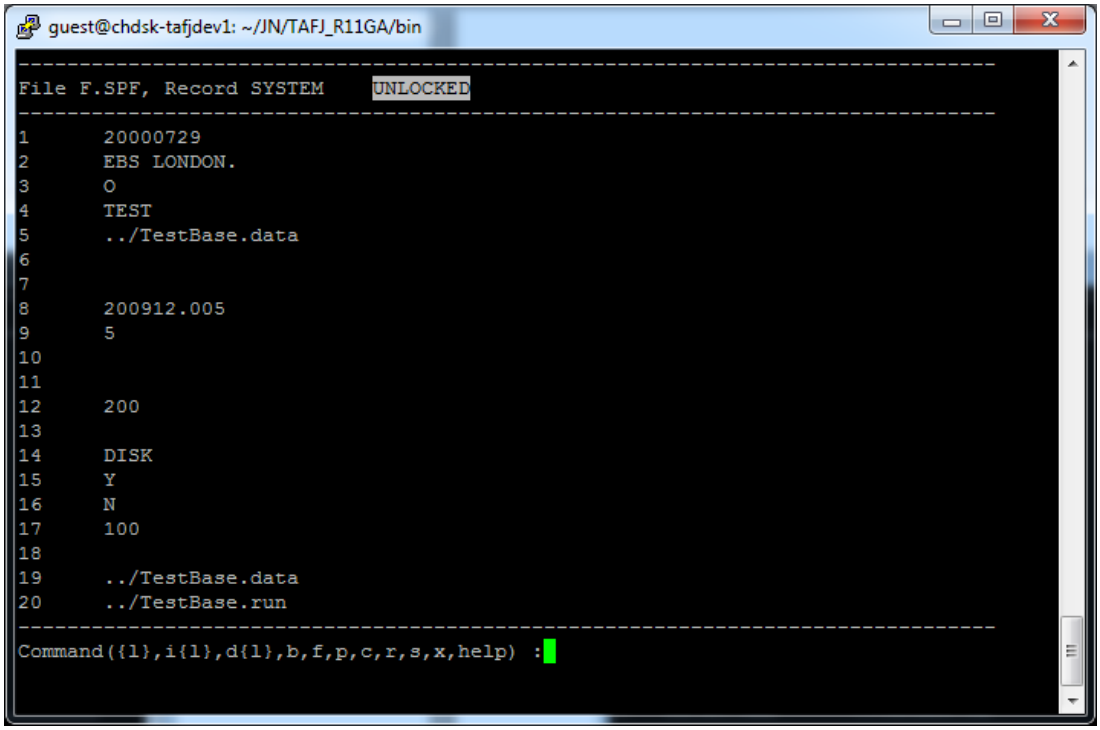
Change column name mode

If you launched JED console with -**cn** option you will be able to enable and disable the field name display in the margin on left part of the screen.

By default field name display will be enabled.

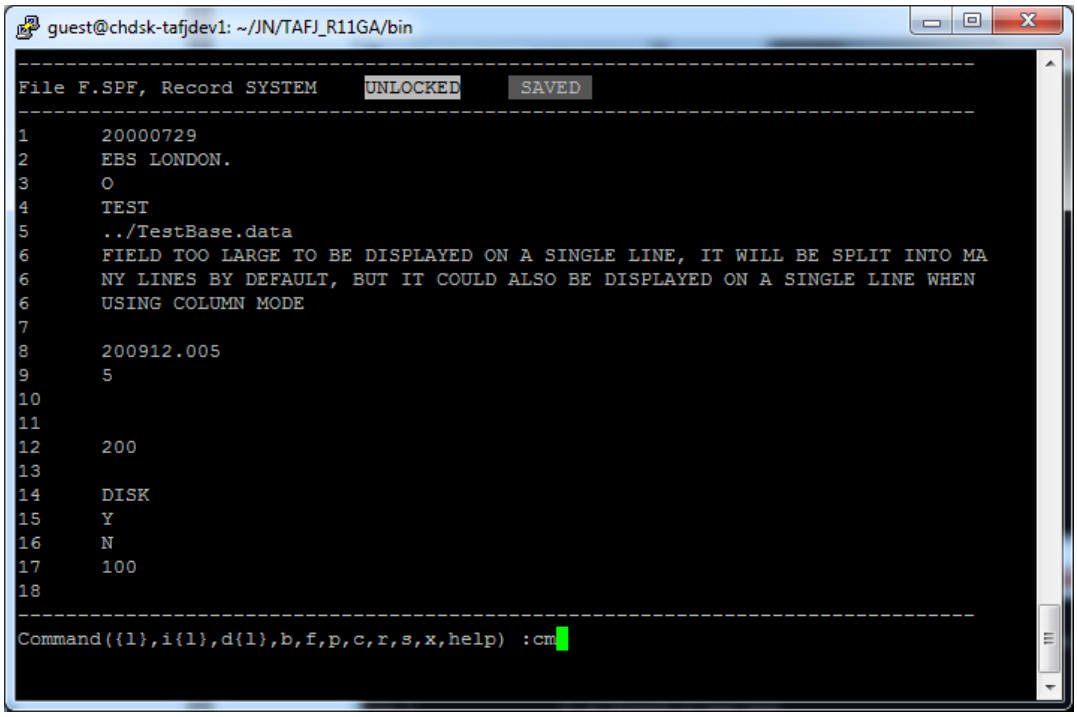
You can disable it by using the **cn** command.




guest@chdsk-tafjdev1: ~/JN/TAFJ_R11GA/bin
File F.SPF, Record SYSTEM UNLOCKED
1 20000729
2 EBS LONDON.
3 O
4 TEST
5 ../TestBase.data
6
7
8 200912.005
9 5
10
11
12 200
13
14 DISK
15 Y
16 N
17 100
18
19 ../TestBase.data
20 ../TestBase.run
Command({l},i{l},d{l},b,f,p,c,r,s,x,help) :

Change column mode

By default JED console will split into as many lines as necessary, fields that are too large to be displayed on a single line.

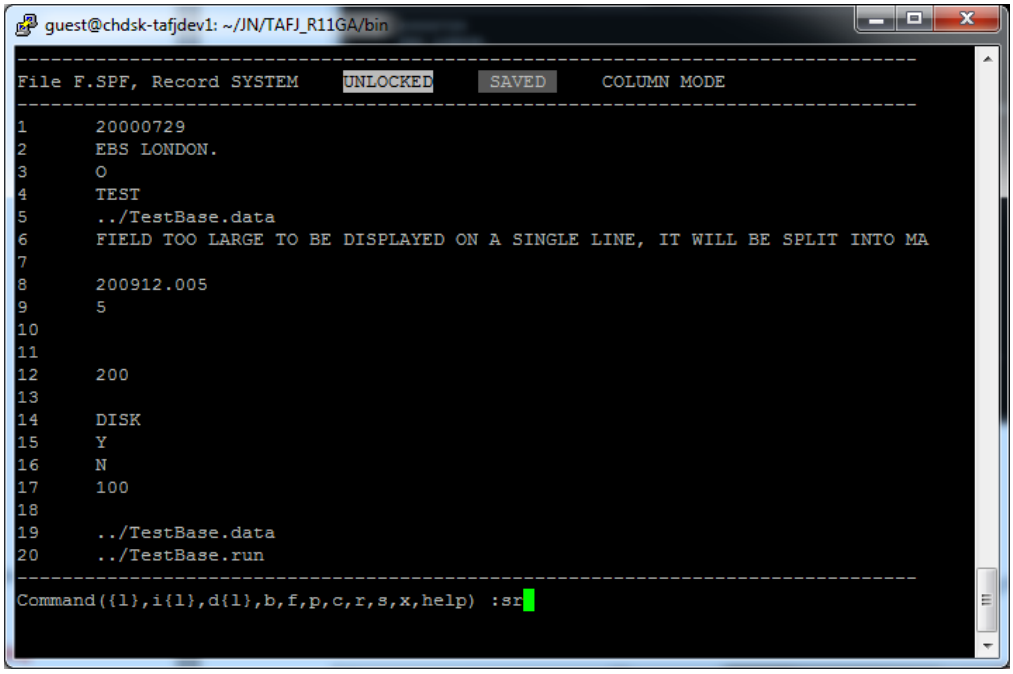


You could then pass into COLUMN MODE with command **cm**, the field will be displayed on a single line and multiple columns.

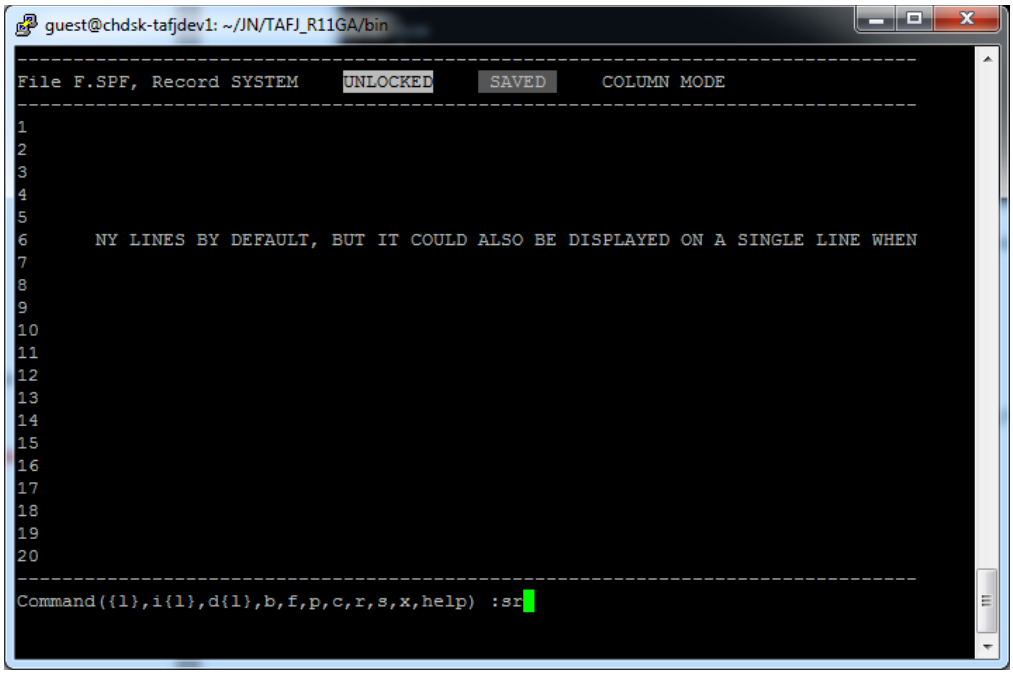
Please be sure to save your pending changes before switching to column mode, as console display will be reinitialized all unsaved modifications will be lost when changing mode.

A flag in top right of the console indicates that column mode is on.

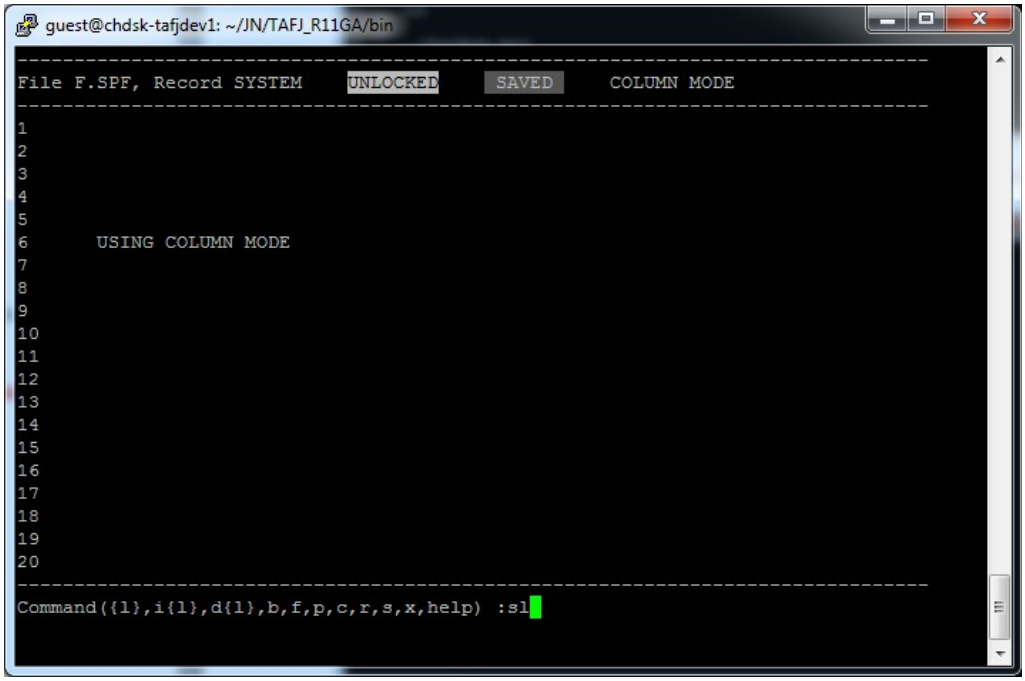
You can now scroll right and left to visualize the entire field, by using commands **sr** and **sl.**



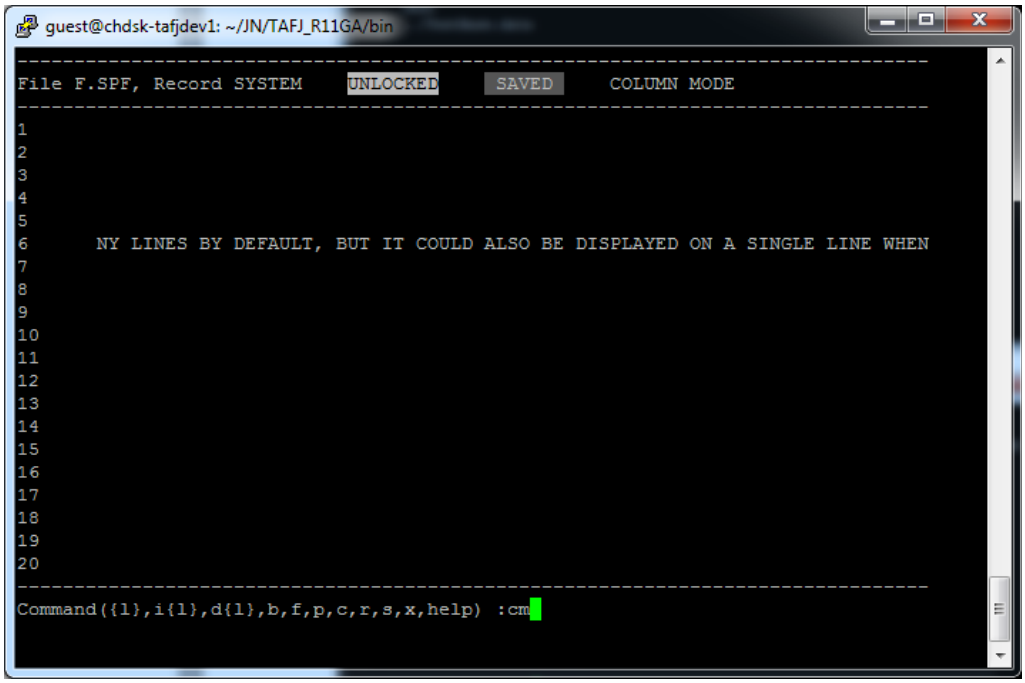
i.e. use command **sr** to scroll right to next column, and so on until the end of field.

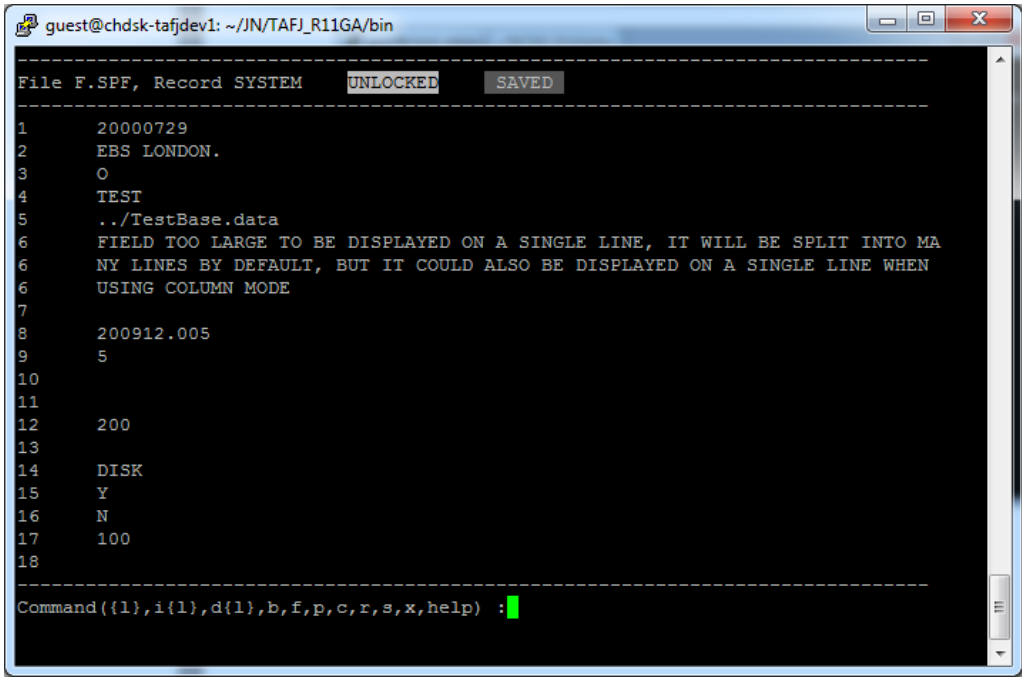


You can then use command **sl** to go back to previous column



By typing **cm** command you will disable the column mode and get back to the classic mode.

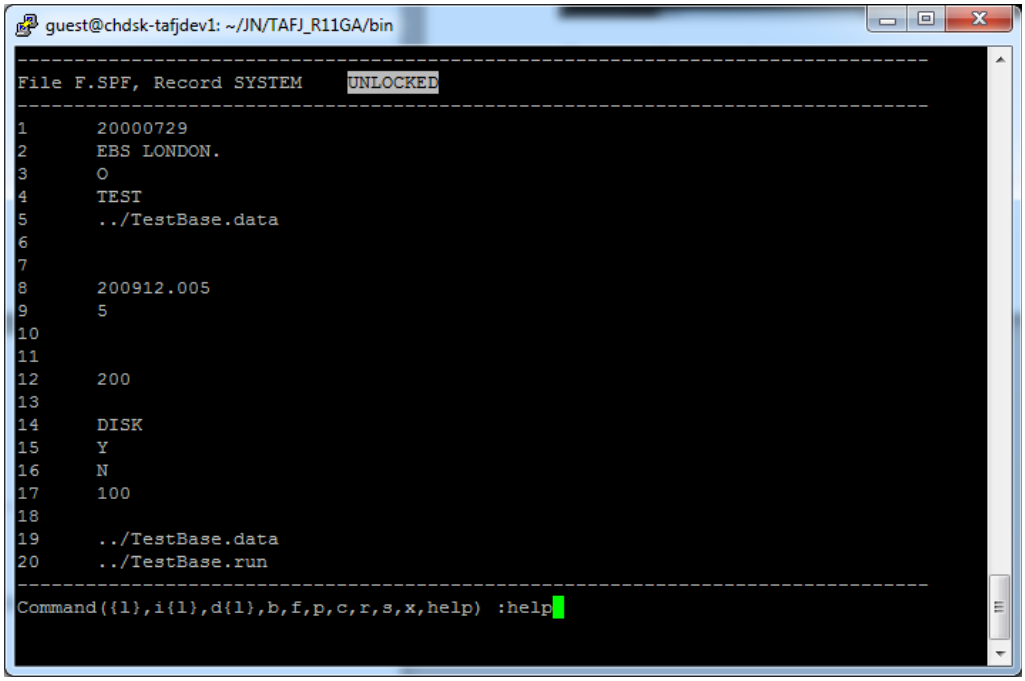


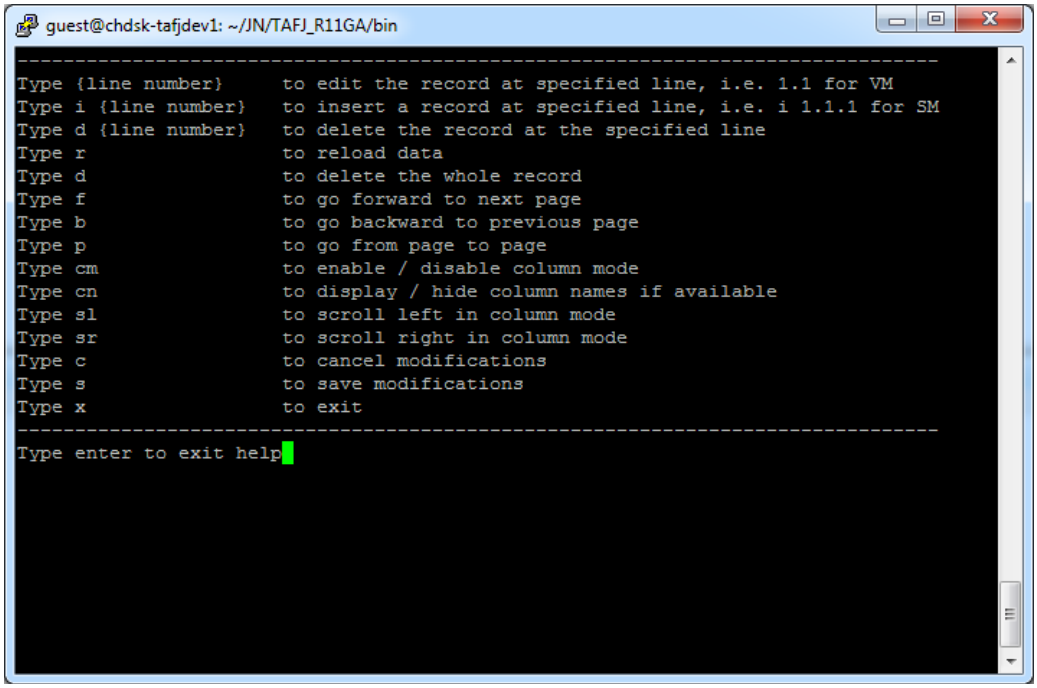


Display help

An integrated help that sum up the available command can be reached by using **help** command.

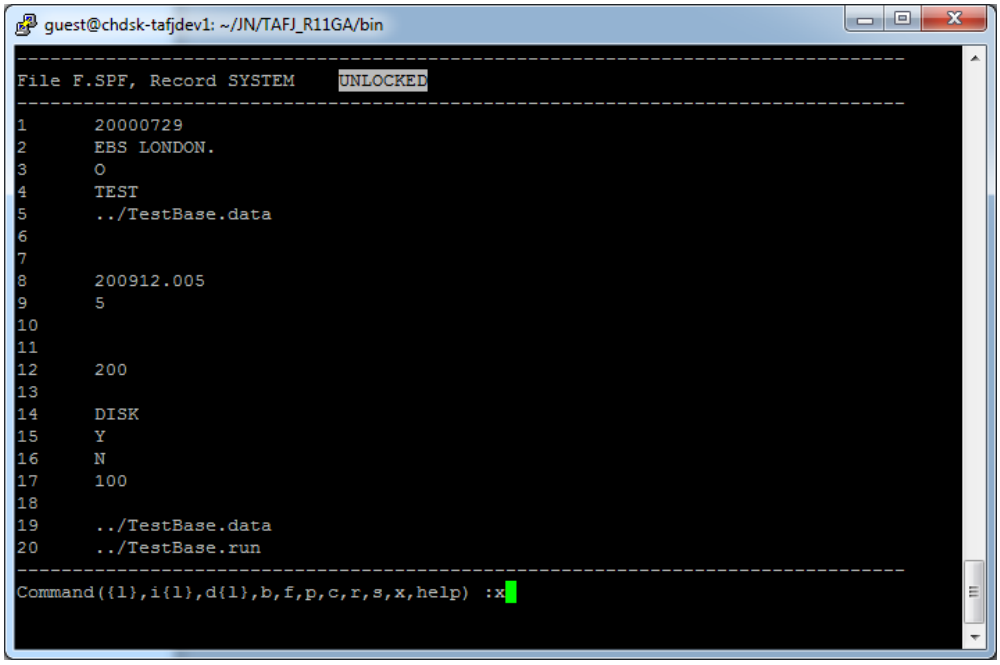
Type “enter” to exit help.





Exit JED Console application

Simply type the exit command **x** to quit. All unsaved modification will be lost.



DBCheck

Introduction

DBCheck is part of the TAFJ-DB tools. It allows checking your database integrity by selecting from the TAFJ_VOC all the tables, and for each, it will do the following:

- Open the table and check its type
- Write a test record on it
- Read that test record
- Read that test record and lock it
- Delete the test record

These operations are being displayed on the console with the format:

Table name - Table type - W(write) R(read) RU (read and lock) D (delete) - Time elapsed

Syntax

The DBCheck syntax is the following:

```
DBCheck [-cf <configuration>[.properties]]
```

The targeted database will be the one defined in your default configuration file or in the configuration specified by the -cf option.

Example

```
DBCheck
```

The resulting output looks like this:

```
_____________________________________DB Check_____________________________________
1) FBNK.AA.ARR.ACTIVITY.API$HIS                 Type =XML   WRRUD  407 ms [DONE ]
2) FEU1.AA.ARR.ACTIVITY.API$NAU                 Type =XML   WRRUD  31 ms  [DONE ]
3) FBNK.AA.ARR.ACTIVITY.CHARGES                 Type =XML   WRRUD  31 ms  [DONE ]
4) FMF1.AA.ARR.ACTIVITY.CHARGES$NAU             Type =XML   WRRUD  47 ms  [DONE ]
5) FSG1.AA.ARR.ACTIVITY.CHARGES$SIM             Type =XML   WRRUD  47 ms  [DONE ]
6) FEU1.AA.ARR.ACTIVITY.MAPPING$NAU             Type =XML   WRRUD  47 ms  [DONE ]
                                  …….
17673) FSG1.SC.FEE.GROUP.HIST                   Type =XML   WRRUD  31 ms  [DONE ]
17674) FSG1.SC.FEE.GROUP.HIST$NAU               Type =XML   WRRUD  16 ms  [DONE ]
17675) FSG1.SC.FEES.MODIFY                      Type =XML   WRRUD  15 ms  [DONE ]
17676) FSG1.SC.FUND.FLOW                        Type =XML   WRRUD  32 ms  [DONE ]
_____________________________________________________________________________
```

DBComparer

Introduction

DBComparer is part of the TAFJ-DB tools. It compares the structure and data from two databases defined in the configuration file: the main database and the comparer. It will stop as soon as it will find a difference and it will indicate it in the console.

Requirements

As for the targeted database, the following properties must be set to define the comparer:

- temn.tafj.jdbc.url.comparer
- temn.tafj.jdbc.driver.comparer
- temn.tafj.jdbc.username.comparer
- temn.tafj.jdbc.password.comparer

The following options are available to customize the process:

-ignorefiles : A comma separated list of files without spaces to ignore in the comparison.

-files : A comma separated list of files without spaces to include (without whole VOC) in the comparison

-startwithfile : All of the files before the file specified in the VOC will be skipped.

-threads : Number of threads to start (default is 1).

Syntax

The DBComparer syntax to perform a full comparison is the following:

DBComparer

Example

To run a comparison that starts at the mentioned file in the VOC.

```
DBComparer –startswithfile FBNK.CURRENCY
```

In case of difference in the structure the resulting output looks like this:

```
_____________________________________DBComparer_____________________________________
Voc Record Key: %DATA.ACC.VOC
Voc Record Key: %DATA.ACCOUNT
Voc Record Key: %DICTIONARY
Voc Record Key: %OTHER.VOC
Unable to correctly select VOC record from comparer database: %OTHER.VOC
```

In case of difference in the data the resulting output looks like this:

```
_____________________________________DBComparer_____________________________________
pool-1-thread-1 file: (-1) F.TT.TELLER.HP.VERSIONS Checking record: 1
pool-1-thread-1 file: (-1) F.TSA.WORKLOAD.PROFILE$HIS Checking record: 1
pool-1-thread-1 file: (-1) F.TSA.WORKLOAD.PROFILE Checking record: 1
java.lang.RuntimeException: DB::SSELECT::Comparer returned count 214 where
other returned count 219
Comparer list:
Other    list: F.TSA.STATUS
cancelling other threads.
```

TAFJ Promoted Columns

This section explains how to promote columns for TAFJ for performance improvements. Promoted columns are relational columns where data is generated via a function when it is inserted/updated. Therefore, these columns will perform better when indexed over an xml index.

This SQL mode command outputs a script to create a promoted column for indexing. Use it with the SPOOL command.

CREATE-PROMOTED-COL <<FUNCTION NAME>> ATTRIBUTE=<<Att# in Dict>> DATATYPE=<<(NUMBER|VARCHAR)>> LENGTH=<<Length for Col>>COLNAME=<<COL to index>> TABLENAME=<<table>>

Here is an example:

tafj@localhost:TB201507>SPOOL

tafj@localhost:TB201507>CREATE-PROMOTED-COL IX_RANK ATTRIBUTE=2 DATATYPE=VARCHAR LENGTH=30 COLNAME=RANK TABLENAME=FBNK_CURRENCY

tafj@localhost:TB201507>SPOOL

The resulting script should show up under $TAFJ_HOME/log/DBTools.

Below are examples of what to do for each database.

Promoting Columns for SQLServer example

```
Below is an example of how to promote the RANK column (xml attribute 1) on the
FBNK_CURRENCY table

set QUOTED_IDENTIFIER on;
go

CREATE FUNCTION udf_RANK_CURRENCY_C1 (@xmlrecord XML)
RETURNS integer
WITH SCHEMABINDING
BEGIN
RETURN @xmlrecord.value('(/row/c1/text())[1]', 'integer')
END

ALTER TABLE FBNK_CURRENCY
ADD RANK AS dbo.udf_RANK_CURRENCY_C1(XMLRECORD) PERSISTED

CREATE INDEX ix_FBNK_CURRENCY_RANK ON FBNK_CURRENCY(RANK)
```

Promoting Columns for DB2 example

drop function extractC1_INT@

```
create function extractC1_INT(xmlrecord XML)
returns INTEGER
language sql contains sql
no external action deterministic
return xmlcast(xmlquery('$t/row/c1' passing xmlrecord as "t") as varchar(10))@

set integrity for FBNK_CURRENCY off@

alter table FBNK_CURRENCY add RANK INTEGER generated always as
(extractC1_INT(XMLRECORD))@

set integrity for FBNK_CURRENCY immediate checked force generated @

create index IX_FBNK_CURRENCY_RANK on FBNK_CURRENCY(RANK)@
```

Promoting Columns for Oracle example

```
alter table fbnk_currency
add (
RANK  number(10)   as (NVL(CAST(extractValue(xmlrecord,'/row/c1') as NUMBER),0))
);

create index IX_FBNK_CURRENCY_RANK on FBNK_CURRENCY(RANK)
```

Drop and Recreate the Views (All Databases)

Drop and recreate view so that the new column will go directly to the rdbms column and not the XML column when the view is queried. (View can be retrieved from the particular database or generated by using DBImport with DBImport logging set to DEBUG in $TAFJ_HOME/conf/TAFJTrace.properties. Here you just want to regenerate the VIEW ONLY for the PARTICULAR TABLE ONLY)

Oracle example of getting the view definition from the database:

```
SQL> set long 100000
SQL> select text from all_views where view_name = 'TAFJV_FBNK_CURRENCY';
```

This is a DB2 example only for recreating the view to use the new column RANK.

```
CREATE VIEW TAFJV_FBNK_CURRENCY as
SELECT a.RECID, a.XMLRECORD "THE_RECORD"
,a.RECID "CURRENCY_CODE"
,RANK "RANK"
,XMLCAST(XMLQUERY('$d/row/c2[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"NUMERIC_CCY_CODE"
,XMLCAST(XMLQUERY('$d/row/c3[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "CCY_NAME"
,XMLQUERY('$d/row/c3' passing a.XMLRECORD as "d") "CCY_NAME_3"
,XMLCAST(XMLQUERY('$d/row/c4[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "NO_OF_DECIMALS"
,XMLCAST(XMLQUERY('$d/row/c5[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "QUOTATION_CODE"
,XMLCAST(XMLQUERY('$d/row/c6[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "QUOTATION_PIPS"
,XMLCAST(XMLQUERY('$d/row/c7[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "DAYS_DELIVERY"
,XMLCAST(XMLQUERY('$d/row/c8[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "DAYS_FORWARD"
,XMLCAST(XMLQUERY('$d/row/c9[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"INTEREST_DAY_BASIS"
,XMLCAST(XMLQUERY('$d/row/c10[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"RATE_ALLOWANCE"
```

```
,XMLCAST(XMLQUERY('$d/row/c11[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "FIXING_DATE"
,XMLCAST(XMLQUERY('$d/row/c12[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"CURRENCY_MARKET"
,XMLQUERY('$d/row/c12' passing a.XMLRECORD as "d") "CURRENCY_MARKET_12"
,XMLCAST(XMLQUERY('$d/row/c13[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"QUOTATION_SUSP"
,XMLQUERY('$d/row/c13' passing a.XMLRECORD as "d") "QUOTATION_SUSP_13"
,XMLCAST(XMLQUERY('$d/row/c14[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "MID_REVAL_RATE"
,XMLQUERY('$d/row/c14' passing a.XMLRECORD as "d") "MID_REVAL_RATE_14"
,XMLCAST(XMLQUERY('$d/row/c15[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"DEFAULT_SPREAD"
,XMLQUERY('$d/row/c15' passing a.XMLRECORD as "d") "DEFAULT_SPREAD_15"
,XMLCAST(XMLQUERY('$d/row/c16[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "BUY_RATE"
,XMLQUERY('$d/row/c16' passing a.XMLRECORD as "d") "BUY_RATE_16"
,XMLCAST(XMLQUERY('$d/row/c17[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "SELL_RATE"
,XMLQUERY('$d/row/c17' passing a.XMLRECORD as "d") "SELL_RATE_17"
,XMLCAST(XMLQUERY('$d/row/c18[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"UPTO_SMALL_AMT"
,XMLQUERY('$d/row/c18' passing a.XMLRECORD as "d") "UPTO_SMALL_AMT_18"
,XMLCAST(XMLQUERY('$d/row/c19[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"TRSY_SMALL_SPRD"
,XMLQUERY('$d/row/c19' passing a.XMLRECORD as "d") "TRSY_SMALL_SPRD_19"
,XMLCAST(XMLQUERY('$d/row/c20[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"CUST_SMALL_SPRD"
,XMLQUERY('$d/row/c20' passing a.XMLRECORD as "d") "CUST_SMALL_SPRD_20"
,XMLCAST(XMLQUERY('$d/row/c21[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "NEGOTIABLE_AMT"
,XMLQUERY('$d/row/c21' passing a.XMLRECORD as "d") "NEGOTIABLE_AMT_21"
,XMLCAST(XMLQUERY('$d/row/c22[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"TRSY_MED_SPREAD"
,XMLQUERY('$d/row/c22' passing a.XMLRECORD as "d") "TRSY_MED_SPREAD_22"
,XMLCAST(XMLQUERY('$d/row/c23[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"CUST_MED_SPREAD"
,XMLQUERY('$d/row/c23' passing a.XMLRECORD as "d") "CUST_MED_SPREAD_23"
,XMLCAST(XMLQUERY('$d/row/c24[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "REVAL_RATE"
,XMLQUERY('$d/row/c24' passing a.XMLRECORD as "d") "REVAL_RATE_24"
,XMLCAST(XMLQUERY('$d/row/c25[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "TRSY_LIMIT_AMT"
,XMLQUERY('$d/row/c25' passing a.XMLRECORD as "d") "TRSY_LIMIT_AMT_25"
,XMLCAST(XMLQUERY('$d/row/c26[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"MIN_ROUND_AMOUNT"
,XMLCAST(XMLQUERY('$d/row/c27[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"CASH_ONLY_ROUNDING"
,XMLCAST(XMLQUERY('$d/row/c28[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"MIN_ROUND_TYPE"
,XMLCAST(XMLQUERY('$d/row/c29[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"CASH_ROUND_TYPE"
,XMLCAST(XMLQUERY('$d/row/c31[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "PRECIOUS_METAL"
,XMLCAST(XMLQUERY('$d/row/c32[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"EQUIVALENT_CCYS"
,XMLQUERY('$d/row/c32' passing a.XMLRECORD as "d") "EQUIVALENT_CCYS_32"
,XMLCAST(XMLQUERY('$d/row/c33[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "COUNTRY_CODE"
,XMLCAST(XMLQUERY('$d/row/c34[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "FIXED_RATE"
,XMLCAST(XMLQUERY('$d/row/c35[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "FIXED_CCY"
,XMLCAST(XMLQUERY('$d/row/c36[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000))
"FIXED_START_DATE"
,XMLCAST(XMLQUERY('$d/row/c37[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "BASE_CCY_RANK"
,XMLCAST(XMLQUERY('$d/row/c38[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "AVAILABLE_DATE"
,XMLCAST(XMLQUERY('$d/row/c39[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "BLOCKED_DATE"
,XMLCAST(XMLQUERY('$d/row/c40[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "CUT_OFF_TIME"
,XMLCAST(XMLQUERY('$d/row/c41[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "CLS_CCY"
,XMLCAST(XMLQUERY('$d/row/c42[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED14"
,XMLCAST(XMLQUERY('$d/row/c43[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED13"
,XMLCAST(XMLQUERY('$d/row/c44[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED12"
,XMLCAST(XMLQUERY('$d/row/c45[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED11"
,XMLCAST(XMLQUERY('$d/row/c46[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED10"
,XMLCAST(XMLQUERY('$d/row/c47[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED9"
,XMLCAST(XMLQUERY('$d/row/c48[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED8"
,XMLCAST(XMLQUERY('$d/row/c49[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED7"
,XMLCAST(XMLQUERY('$d/row/c50[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED6"
,XMLCAST(XMLQUERY('$d/row/c51[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED5"
,XMLCAST(XMLQUERY('$d/row/c52[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED4"
,XMLCAST(XMLQUERY('$d/row/c53[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED3"
```

```
,XMLCAST(XMLQUERY('$d/row/c54[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED2"
,XMLCAST(XMLQUERY('$d/row/c55[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RESERVED1"
,XMLCAST(XMLQUERY('$d/row/c56[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "OVERRIDE"
,XMLQUERY('$d/row/c56' passing a.XMLRECORD as "d") "OVERRIDE_56"
,XMLCAST(XMLQUERY('$d/row/c57[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "RECORD_STATUS"
,XMLCAST(XMLQUERY('$d/row/c58[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "CURR_NO"
,XMLCAST(XMLQUERY('$d/row/c59[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "INPUTTER"
,XMLQUERY('$d/row/c59' passing a.XMLRECORD as "d") "INPUTTER_59"
,XMLCAST(XMLQUERY('$d/row/c60[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "DATE_TIME"
,XMLQUERY('$d/row/c60' passing a.XMLRECORD as "d") "DATE_TIME_60"
,XMLCAST(XMLQUERY('$d/row/c61[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "AUTHORISER"
,XMLCAST(XMLQUERY('$d/row/c62[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "CO_CODE"
,XMLCAST(XMLQUERY('$d/row/c63[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "DEPT_CODE"
,XMLCAST(XMLQUERY('$d/row/c64[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "AUDITOR_CODE"
,XMLCAST(XMLQUERY('$d/row/c65[position()=1]' passing a.XMLRECORD as "d") as VARCHAR(4000)) "AUDIT_DATE_TIME"
FROM
"FBNK_CURRENCY" a
```

USE CASE

Here is an advanced example with column SCRIPT.GROUP a multi-value column

Jql

```
SELECT F.SEAT.SCRIPTS WITH SCRIPT.STATUS EQ 'ACTIVE' AND WITH SCRIPT.GROUP EQ 'TB01-START'
```

Sql translation

```
SELECT RECID FROM "TAFJV_F_SEAT_SCRIPTS" WHERE "SCRIPT_STATUS" = 'ACTIVE' and
( XMLEXISTS('$t/c10[text()="TB01-START"]' PASSING "SCRIPT_GROUP_10" as "t") )
```

The above creates a full table scan. Here is one way to index it.

Alter Table for each multi-value

```
ALTER TABLE F_SEAT_SCRIPTS ADD( SCRIPT_GROUP_1  VARCHAR2(4000) as
( substr(extractValue(xmlrecord,'/row/c10[1]'),1,100) ));
ALTER TABLE F_SEAT_SCRIPTS ADD( SCRIPT_GROUP_2  VARCHAR2(4000) as
( substr(extractValue(xmlrecord,'/row/c10[2]'),1,100) ));
ALTER TABLE F_SEAT_SCRIPTS ADD( SCRIPT_GROUP_3  VARCHAR2(4000) as
( substr(extractValue(xmlrecord,'/row/c10[3]'),1,100) ));
ALTER TABLE F_SEAT_SCRIPTS ADD( SCRIPT_GROUP_4  VARCHAR2(4000) as
( substr(extractValue(xmlrecord,'/row/c10[4]'),1,100) ));
ALTER TABLE F_SEAT_SCRIPTS ADD( SCRIPT_GROUP_5  VARCHAR2(4000) as
( substr(extractValue(xmlrecord,'/row/c10[5]'),1,100) ));
```

Create Indexes for each new column

```
CREATE INDEX ix_f_seat_scripts_script_grp_1 ON F_SEAT_SCRIPTS(SCRIPT_GROUP_1);
CREATE INDEX ix_f_seat_scripts_script_grp_2 ON F_SEAT_SCRIPTS(SCRIPT_GROUP_2);
CREATE INDEX ix_f_seat_scripts_script_grp_3 ON F_SEAT_SCRIPTS(SCRIPT_GROUP_3);
CREATE INDEX ix_f_seat_scripts_script_grp_4 ON F_SEAT_SCRIPTS(SCRIPT_GROUP_4);
CREATE INDEX ix_f_seat_scripts_script_grp_5 ON F_SEAT_SCRIPTS(SCRIPT_GROUP_5);
CREATE INDEX ix_f_seat_scripts_script_grp_n ON F_SEAT_SCRIPTS (SCRIPT_GROUP_N);
```

Recreate the view with new columns

```
CREATE OR REPLACE VIEW TAFJV_F_SEAT_SCRIPTS as SELECT a.RECID, a.XMLRECORD "THE_RECORD"
,a.RECID "SCRIPT_ID"
,extractValue(a.XMLRECORD,'/row/c1[position()=1]') "DESCRIPT"
,extract(a.XMLRECORD,'/row/c1') "DESCRIPT_1"
,extractValue(a.XMLRECORD,'/row/c2[position()=1]') "COMPANY_CODE"
,extractValue(a.XMLRECORD,'/row/c3[position()=1]') "SCRIPT_STATUS"
,extractValue(a.XMLRECORD,'/row/c4[position()=1]') "SCRIPT_SOURCE"
,extractValue(a.XMLRECORD,'/row/c5[position()=1]') "ALTERNATE_REF"
,extractValue(a.XMLRECORD,'/row/c6[position()=1]') "SELECT_ROUTINE"
,extractValue(a.XMLRECORD,'/row/c7[position()=1]') "BASE_RELEASE"
,extractValue(a.XMLRECORD,'/row/c8[position()=1]') "PRODUCT_GROUP"
,extractValue(a.XMLRECORD,'/row/c9[position()=1]') "PRODUCT_CODE"
,extractValue(a.XMLRECORD,'/row/c10[position()=1]') "SCRIPT_GROUP"
,SCRIPT_GROUP_N "SCRIPT_GROUP_10"
,SCRIPT_GROUP_1
,SCRIPT_GROUP_2
,SCRIPT_GROUP_3
,SCRIPT_GROUP_4
,SCRIPT_GROUP_5
,extractValue(a.XMLRECORD,'/row/c11[position()=1]') "APPLICATION"
,extractValue(a.XMLRECORD,'/row/c12[position()=1]') "STATIC_SETUP"
,extractValue(a.XMLRECORD,'/row/c13[position()=1]') "VERSION"
```

```
,extractValue(a.XMLRECORD,'/row/c14[position()=1]') "FUNCTION"
,extractValue(a.XMLRECORD,'/row/c15[position()=1]') "TXN_ID"
,extractValue(a.XMLRECORD,'/row/c16[position()=1]') "FIELD_NAME"
,extract(a.XMLRECORD,'/row/c16') "FIELD_NAME_16"
,extractValue(a.XMLRECORD,'/row/c17[position()=1]') "FIELD_VALUE"
,extract(a.XMLRECORD,'/row/c17') "FIELD_VALUE_17"
,extractValue(a.XMLRECORD,'/row/c18[position()=1]') "FIELD_INPUT"
,extract(a.XMLRECORD,'/row/c18') "FIELD_INPUT_18"
,extractValue(a.XMLRECORD,'/row/c19[position()=1]') "SEAT_ID"
,extractValue(a.XMLRECORD,'/row/c20[position()=1]') "UPDATE_APPL"
,extract(a.XMLRECORD,'/row/c20') "UPDATE_APPL_20"
,extractValue(a.XMLRECORD,'/row/c21[position()=1]') "UPDATE_SCRPT_ID"
,extract(a.XMLRECORD,'/row/c21') "UPDATE_SCRPT_ID_21"
,extractValue(a.XMLRECORD,'/row/c22[position()=1]') "UPD_APPL_FLD_IDEN"
,extract(a.XMLRECORD,'/row/c22') "UPD_APPL_FLD_IDEN_22"
,extractValue(a.XMLRECORD,'/row/c23[position()=1]') "GENERATE_ERROR"
,extractValue(a.XMLRECORD,'/row/c24[position()=1]') "DEFINE_ERROR"
,extractValue(a.XMLRECORD,'/row/c25[position()=1]') "CREATED_BY"
,extractValue(a.XMLRECORD,'/row/c26[position()=1]') "APPROVED_BY"
,extractValue(a.XMLRECORD,'/row/c27[position()=1]') "USER"
,extractValue(a.XMLRECORD,'/row/c28[position()=1]') "MSG_FROM_FILE"
,extractValue(a.XMLRECORD,'/row/c29[position()=1]') "TPR_FILE_NAME"
,extractValue(a.XMLRECORD,'/row/c30[position()=1]') "CREATED_DATE"
,extractValue(a.XMLRECORD,'/row/c31[position()=1]') "LAST_MODIFIED_DATE"
,extractValue(a.XMLRECORD,'/row/c32[position()=1]') "IO_ANALYSIS"
,extract(a.XMLRECORD,'/row/c32') "IO_ANALYSIS_32"
,extractValue(a.XMLRECORD,'/row/c33[position()=1]') "RESERVED_2"
,extractValue(a.XMLRECORD,'/row/c34[position()=1]') "RESERVED_1"
,extractValue(a.XMLRECORD,'/row/c36[position()=1]') "OVERRIDE"
,extract(a.XMLRECORD,'/row/c36') "OVERRIDE_36"
,extractValue(a.XMLRECORD,'/row/c37[position()=1]') "RECORD_STATUS"
,extractValue(a.XMLRECORD,'/row/c38[position()=1]') "CURR_NO"
,extractValue(a.XMLRECORD,'/row/c39[position()=1]') "INPUTTER"
,extract(a.XMLRECORD,'/row/c39') "INPUTTER_39"
,extractValue(a.XMLRECORD,'/row/c40[position()=1]') "DATE_TIME"
,extract(a.XMLRECORD,'/row/c40') "DATE_TIME_40"
,extractValue(a.XMLRECORD,'/row/c41[position()=1]') "AUTHORISER"
,extractValue(a.XMLRECORD,'/row/c42[position()=1]') "CO_CODE"
,extractValue(a.XMLRECORD,'/row/c43[position()=1]') "DEPT_CODE"
,extractValue(a.XMLRECORD,'/row/c44[position()=1]') "AUDITOR_CODE"
,extractValue(a.XMLRECORD,'/row/c45[position()=1]') "AUDIT_DATE_TIME"
FROM
"F_SEAT_SCRIPTS" a
```

Change query

```
SELECT RECID FROM TAFJV_F_SEAT_SCRIPTS WHERE "SCRIPT_STATUS" =
'ACTIVE' and (SCRIPT_GROUP_1 = 'TB01-START' OR SCRIPT_GROUP_2 = 'TB01-
START' OR SCRIPT_GROUP_3 = 'TB01-START' OR SCRIPT_GROUP_4 = 'TB01-START'
OR SCRIPT_GROUP_5 = 'TB01-START');
```