



jBASE Data Provider

Version 4.1



Copyright

Copyright (c) 2007 TEMENOS HOLDINGS NV

All rights reserved.

This document contains proprietary information that is protected by copyright. No part of this document may be reproduced, transmitted, or made available directly or indirectly to a third party without the express written agreement of TEMENOS UK Limited. Receipt of this material directly from TEMENOS UK Limited constitutes its express permission to copy. Permission to use or copy this document expressly excludes modifying it for any purpose, or using it to create a derivative therefrom.

Acknowledgements

Information regarding Unicode has been provided in part courtesy of the Unicode Consortium. The Unicode Consortium is a non-profit organization founded to develop, extend and promote use of the Unicode Standard, which specifies the representation of text in modern software products and standards. The membership of the consortium represents a broad spectrum of corporations and organizations in the computer and information processing industry. The consortium is supported financially solely through membership dues. Membership in the Unicode Consortium is open to organizations and individuals anywhere in the world who support the Unicode Standard and wish to assist in its extension and implementation.

Portions of the information included herein regarding IBM's ICU has been reprinted by permission from International Business Machines Corporation copyright 2001 jBASE, jBASE BASIC, jED, jSHELL, jLP, jEDI, jCL, jQL, j3 j4 and jPLUS files are trademarks of TEMENOS Holdings NV.

REALITY is a trademark of Northgate Solutions Limited.

PICK is a trademark of Raining Data Inc.

All other trademarks are acknowledged.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows, Windows NT, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names used in this publication may be trademarks or service marks of others.

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:

Technical Publications Department

TEMENOS UK Limited

2 PeopleBuilding

Hemel Hempstead

Hertfordshire

HP2 6NW

England

Tel SB: +44 (0) 1442 431000

Fax: +44 (0) 1442 431001

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

Contents

Documentation Conventions	1
Limitations of this documentation.....	3
WHAT IS JDP?	4
OVERVIEW OF AVAILABLE MODES	4
Mode 1	4
Mode 2	5
Mode 3	5
GETTING STARTED.....	7
What should be installed by default	7
What environment variables are needed?	7
How do I start the irpcd demon?	8
Description of scripts & files.....	9
UPGRADING THE JDP ENVIRONMENT.....	10
THE ADDON FILE	10
NAV_UTIL.....	11
What is NAV_UTIL.....	11
Example of commands	11
EXECUTING SQL	13
SETTING UP JDP FOR EACH SUPPORTED MODE	14
Standard datasource.....	15
TABLEFILE datasource.....	19
Catalogue datasource.....	24
JBASE DICTIONARY CONFIGURATOR.....	27
JDP AND SUBVALUES	29

Documentation Conventions

This manual uses the following conventions:

Convention	Usage
BOLD	In syntax, bold indicates commands, function names, and options. In text, bold indicates keys to press, function names, menu selections, and MS-DOS commands.
UPPERCASE	In syntax, uppercase indicates JBASE commands, keywords, and options; BASIC statements and functions; and SQL statements and keywords. In text, uppercase also indicates JBASE identifiers such as filenames, account names, schema names, and Windows NT filenames and pathnames.
UPPERCASE <i>Italic</i>	In syntax, italic indicates information that you supply. In text, italic also indicates UNIX commands and options, filenames, and pathnames.
Courier	Courier indicates examples of source code and system output.
Courier Bold	Courier Bold In examples, courier bold indicates characters that the user types or keys (for example, <Return>).
[]	Brackets enclose optional items. Do not type the brackets unless indicated.
{ }	Braces enclose nonoptional items from which you must select at least one. Do not type the braces.
ItemA itemB	A vertical bar separating items indicates that you can choose only one item. Do not type the vertical bar.
...	Three periods indicate that more of the same type of item can optionally follow.
⇒	A right arrow between menu options indicates you should choose each option in sequence. For example, “Choose File ⇒ Exit ” means you should choose File from the menu bar, and then choose Exit from the File pull-down menu.

Syntax definitions and examples are indented for ease in reading.

All punctuation marks included in the syntax—for example, commas, parentheses, or quotation marks—are required unless otherwise indicated.

Syntax lines that do not fit on one line in this manual are continued on subsequent lines.

The continuation lines are indented. When entering syntax, type the entire syntax entry, including the continuation lines, on the same input line.

Limitations of this documentation

Because Attunity is a 3rd party product this manual is intended to be nothing more than an introduction to the functionality that is available using Attunity. If you require more detailed examples please read the Attunity manual.

WHAT IS JDP?

jDP is the jBASE driver for the attunity connect software, this provides an interface for the attunity software to access jBASE data files.

Using attunity, any products adhering to the ODBC, OLE DB, JDBC or ADO standards can retrieve data from the jEDI interface (directly from the jBASE database)

Its key features are as follows:

- Allows connectivity to jBASE files via, ODBC \ ADO \ OLDB \ XML type technologies
- Allows applications (front end) to use SQL (Standard Query Languages) syntax
- Standard jBASE files, (back End) need no modification
- Multi-values, Sub-values are supported, and can be grouped together as associated sets.
- Attunity is available on all supported jBASE platforms except iSeries and zSeries.

Overview of available modes

JDP supports three distinct modes of providing access to files:

Mode 1	All visible files mode
Mode 2	TableFile mode
Mode 3	Catalog mode

Mode 1

This is the default mode and allows access to all the files, a user running the jDP driver can open files that:

Can be opened without specifying a path

i.e. are in the user's MD,HOME Dir or JEDIFILEPATH.

Have filenames that are valid SQL table names and have dictionaries.

This mode is useful for ad hoc queries, or for users using shrink-wrapped client software

Mode 2

This mode restricts the view of database files to only those specified in a 'TABLEFILE'. (a TABLEFILE is a hash file, which contains an item for each data file you want to make 'visible').

When configured this will override Mode 1 file searches

This method allows access to be restricted to a nominated set of files by using a special TABLEFILE.

Mode 3

A 'Catalogue' described simply as a 'Database' – a collection of tables that hold data and belong to the database – the database acting as a logical boundary. Unlike the other two methods of accessing the data through jDP, this method allows a good security and permissions strategy to be set up on all the individual files in the database, thus allowing various users access to only specified data.

For the catalogue to maintain a list of users, schemas, tables and permissions, a catalogue directory is set up to hold 'system files'. These system files hold all the needed information that to administer the catalogue.

Within a catalogue, it is possible to create 'Schemas': described as workspaces where users can place\attach their tables. If no Schemas are used, the catalogue will use the public schema (this is the default schema). Schemas are used to hold files for individual users (e.g., Bob) or a logical group of users (e.g. SalesTeam). Each individual schema must have its own working directory where it stores its tables.

When referencing a table in a catalogue, the full explicit reference would be:

CatalogName:SchemaName.TableName

Example Catalogue:

Database \ Catalogue name : NutsAndBoltsPlc

Contains four schemas:

NutsAndBoltsPlc:Public

NutsAndBoltsPlc:SalesTeam

NutsAndBoltsPlc:AdminStaff

NutsAndBoltsPlc:TheMD

With in the SalesTeam schema you may find the following tables:

NutsAndBoltsPlc:SalesTeam.SalesPersonnel

NutsAndBoltsPlc:SalesTeam.Customers

NutsAndBoltsPlc:SalesTeam.SalesOutstanding

NutsAndBoltsPlc:SalesTeam.SalesMade

Getting Started

What should be installed by default

By default a full and working version of jDP should be installed into

`$JBASERELEASEDIR/jdp` (UNIX)
`%JBASERELEASEDIR%\jdp` (WINDOWS)

when you install 4.1.

Within the `jdp` directory there should be all that is needed to run the jDP demo, however if you have not installed jBASE into its default location,

`C:\jbase4\4.1` (Windows)
`/opt/jbase4/4.1` (UNIX)

there may be some additional scripts to be updated...

What environment variables are needed?

In order for jDP to function correctly you need to ensure that both jBASE and jDP are configured correctly,

jDP requires the following :-

On UNIX system

`$NAVROOT`

It should point to the location where jDP has been installed (default `/opt/jbase4/4.1/jdp`)

`$PATH`

Should contain `$NAVROOT/bin`

\$LD_LIBRARY_PATH / \$LIBPATH / \$SHLIBPATH (depending on system)

Should contain \$NAVROOT/lib, as well as any additional subroutines that are needed by your dictionaries.

On WINDOWS system

%NAVROOT%

Should point to the location where jDP has been installed (default C:\jBASE4\4.1\jDP)

%PATH%

Should contain %NAVROOT%\bin

%JBCOBJECTLIST%

Should contain the location of any additional subroutines that are needed by your dictionary's

How do I start the irpcd demon?

On both UNIX and windows systems if you are having problems starting the irpcd demon there should be a description of the last error in one of the log files in \$NAVROOT/tmp.

Generally most common errors are down to the environment not being set correctly, a good test is to open a "jshell" using the same environment that you are using for jDP and see if you can list your files.

On Windows systems use "services" in start >> settings >> control panel >> administrative tools >> Services

On UNIX systems you can use the irpcd.ctl script in \$NAVROOT/bin

As root in \$NAVROOT/bin.

./irpcd.ctl start

Description of scripts & files

On WINDOWS system if you have not installed jBASE to its default location, or on UNIX systems if you are not using a symbolic link to /usr/jbc, you will have to amend to following scripts :-

UNIX

nav_sever.ksh	Sets up the environment when the server is started
site_nav_login.sh	Gets ran for each client session.

WINDOWS

nav_login.bat	Gets ran for each client session.
---------------	-----------------------------------

These scripts are needed to setup the environment for jDP when it first launches a client or server, don't forget that jDP will need to see the same sort of things that jBASE does.

For example you may use subroutines from within your dictionaries or using a different MD for jDP users, in both these example's jDP will fail if you have not setup your jBASE environment variables in the relevant script.

These scripts can be found in \$NAVROOT/bin, and should have a working example that points to the default jBASE/jDP locations.

NOTE: In the \$NAVROOT/def directory there should be:

addon.def	This file is needed to tell the Attunity driver how to connect to jDP.
license.pak	These are the default demo licence supplied by Attunity.
license.txt	

Upgrading the jDP Environment

The installation procedure of jDP gives you the option to upgrade all elements of your existing jDP system. If you decline this automatic comprehensive upgrade, you can later use the supplied upgrade utility to upgrade selected elements of your existing jDP system.

The upgrade utility transfers existing elements of your jDP configuration into object store, jDPs new internal storage mechanism. This utility transfers into object store the following elements in your jDP system:

- jDP metadata
- User profile information specified in the jDP security file
- Stored query specifications (for both procedures and views)
- The results of all nav_util upgrade operations are written to a log residing in either the default location (\tmp under NAVROOT) or the directory specified in the CTL_TRACE_DIR parameter in the jDP environment file.

The ADDON file

The ADDON file is used for the following:

- When connecting to a data source using the ODBC driver on a non-Windows platform.
- When using the data connector and user defined data types SDK. For details see jDP Open Data Connectivity and the Developer SDK.

NAV_UTIL

What is NAV_UTIL

NAV_UTIL is a collection of Attunity utilities. The utilities include troubleshooting utilities and metadata utilities. All of the utilities run from NAV_UTIL. NAV_UTIL can be used for any thing from creating a datasource, running a SQL query or even checking on the status of any running demons.

Example of commands

The following are examples of useful commands, for a more detailed description please read the Attunity documentation.

Adding a new datasource

```
NAV_UTIL UPD_DS DEMO jDP jDP
```

Generating metadata

```
NAV_UTIL GEN_ARRAY_TABLES DEMO *           (For all tables)
NAV_UTIL UPDATE DEMO *                     (For all tables)
```

Or

```
NAV_UTIL GEN_ARRAY_TABLES DEMO JCUSTOMERS ( single file )
NAV_UTIL UPDATE DEMO JCUSTOMERS          ( single file )
```

Adding a remote machine to a client

```
NAV_UTIL EDIT MACHINE
```

Now put an entry in the remoteMachines section

```
<remoteMachines name='NAV' >  
    <remoteMachine name='10_48_3_102' address='10.48.3.102'/>  
</remoteMachines>
```

Next add an entry on the client

```
NAV_UTIL UPD_DS DEMO REMOTE jDP 10_48_3_102'
```

Checking the Status of the Daemon

```
NAV_UTIL check irpcdstat(daemon [, username, password])
```

Viewing what datasources are available

```
NAV_UTIL VIEW MACHINE
```

Executing a SQL Query

```
NAV_UTIL EXECUTE command.sql
```

Where command.sql is a text file containing a valid SQL query

Listing Machines that Have an Active Daemon

```
NAV_UTIL CHECK NETWORK
```

Executing SQL

You can write and execute SQL in the NavSQL environment, as follows:

On-the-fly: Write a sql statement and end it with a semi-colon. Press Enter to execute the statement.

If the SQL contains data from more than one data source, use a colon (:) to identify the data source (that is, `datasource_name:Table_name`).

From a file: Enter the full name of a file that contains sql, prefixed by @. Press Enter to execute the sql contained in the file. For example:

```
NavSQL> @C:\sql\sql-query.sql;
```

will execute the sql contained in the file `sql-query.sql`.

MVS platforms: Use single quotes (') around the filename.

For example, `@'.NAVROOT.TMP.SQL1'`.

You can access the NavSQL environment and run a file immediately by entering the following command:

```
nav_util execute data_source file
```

where `data_source` is the name of the data source as defined in the binding file and `file` is the name of the SQL file.

If you want to run all the queries in the file without the overhead of displaying query information on the screen for each query, enter the following command:

```
nav_util execute data_source -quiet file
```

In this case, only queries that fail cause information to be displayed to the screen during the run. A message is displayed after all the queries have been run, stating the number of queries that succeeded and the number that failed.

From within a transaction: Enter the command `begin-transaction` (optionally with either `read-only` or `write` permission) to start a transaction where you can commit a number of sql statements

together. Use commit to update the data sources with any changes or rollback if you decide that you do not want to accept the changes.

Setting up jDP for each supported mode

In the introduction to this document we mentioned that jDP can be ran in 3 different modes, the following are typical examples of each mode,

Note: \$NAVROOT is the path to where jDP has been installed

\$JBCRELEASEDIR is the path to where jBASE has been installed

\$JEDIFILEPATH is a search path for jBASE to locate files...

\$HOME is the location of you data account,

All of the above should be setup before you start...

Standard datasource

This example shows how to use the standard jBASE environment with jDP, in our sample account we have 3 files

File	Dict	Record position	Description
ORDERS	KEY	0	order ID
	ITEMID1		foreign key to item details
	CUSTID	2	foreign key, customer details
ITEMS	ITEMID0		ID of this item
	DESC	1	Description of item
CUSTOMER	CUSTID	0	Customer ID
	NAME	1	Name of the customer
	ADDR	2	customer's address
CUSTOMERS	F Pointer to CUSTOMER		

The account is held in the directory

WINDOWS: C:\Sample

UNIX: /home/Sample

It also has its own MD, but this is just to demonstrate that it is actually can be used, create a file pointer called CUSTOMERS.

- a. Create the sample tables and dictionaries, and then populate them with some meaningful data.
- b. Adding a datasource.

You will need to let both the client and server know what type of datasource you are creating, which can be done by using one of the following:

“Attunity Configuration Manager”, this is a GUI interface installed with the windows version of Attunity, it also allows you to configure remote servers, (UNIX and WINDOWS)

“nav_util”

nav_util is a multi-functional utility supplied by attunity, it allows jDP to setup and maintain datasources, and it can also execute queries.

To add a datasource via nav_util execute the following from the server.

```
nav_util UPD_DS SAMPLE jDP jDP
```

You can view existing bindings with:-

```
nav_util view bindings
```

For more information please read the Attunity documentation.

c. Setting up the environment variables for a client...

When a client connects the attunity driver needs to be able to see where the jBASE account is.

This is done by using the following script.

```
WINDOWS:  nav_login.bat
UNIX:     site_nav_login.sh
```

WINDOWS:

```
Using jed / notepad edit %NAVROOT%\bin\ nav_login.bat
```

```
jed %NAVROOT%\bin\ nav_login.bat
```

```
File C:\jbase4\4.1\jDP\bin\ , Record 'nav_login.bat' Insert
```

```
17:01:52
```

```
Command->
```

```
001 @echo off
```

```
002 SET PATH="C:\jbase4\4.1\jDP\BIN";%PATH%
```

```
003 SET NAVROOT=C:\jbase4\4.1\jDP
```

```
----- End Of Record -----
```

```
-----
```

Add the following lines to let jBASE know where things are:

```
004 SET JEDIFILEPATH=C:\Sample;.
005 SET JEDIFILENAME_MD=c:\Sample\MD
006 SET HOME=c:\Sample
```

UNIX:

```
jed $NAVROOT/bin/site_nav_login.sh

001 #
002 # Modify Attunity 3.2 environment for jBASE 3.x
003 #
...
035 # Make sure Attunity can see our demo data files
036 JEDIFILEPATH=/usr/jbc/jdp/demo
037 export JEDIFILEPATH
038
039 #
040 # Any additional site specific changes should follow
041
#####
##
```

Add the following lines to let jBASE know where things are..

```
036 JEDIFILEPATH=/home/Sample
037 export JEDIFILEPATH
038 JEDIFILENAME_MD=/home/Sample
039 export JEDIFILENAME_MD
040 HOME=/home
041 export HOME
```

On the client machines you also need to add a reference to the remote datasource, this can be done via the GUI or by executing the following command :

```
nav_util UPD_DS REMOTE <<datasource>>
```

EXAMPLE

```
nav_util UPD_DS REMOTE SAMPLE
```

d) Creating a link between the attunity driver and the jBASE files.

If you are using a file that contains multivalues you will need to configure jDP so that the jDP driver treats the multivalued columns as tables, physically they are still stored as multivalues in the jBASE files, jDP simply creates an internal virtual table that the jDP driver can use.

This is achieved by using the “nav_util” utility,

The general; syntax is as follows...

```
nav_util gen_array_table <<datasource>> <<tablename>>
```

EXAMPLE

```
nav_util gen_array_table SAMPLE CLIENT
nav_util gen_array_table SAMPLE *
```

You can list all data sources with the following command :-

```
nav_util view datasources *
```

The file viewer will vary depending on whether you are running on windows or UNIX in this example we are not using multi-values so all tables should be usable via ODBC without any problems.

For further information see your Attunity documentation.

TABLEFILE datasource

This example shows how to use a TABLEFILE datasource with jDP,

We can use the files that were created for the previous example:

File	Dict	Record position	Description
------	------	-----------------	-------------

ORDERS	KEY	0	order ID
	ITEMID1		foreign key to item details
	CUSTID	2	foreign key to customer details
ITEMS	ITEMID0		ID of this item
	DESC	1	Description of item
CUSTOMER	CUSTID	0	Customer ID
	NAME	1	Name of the customer
	ADDR	2	customer's address

The account is held in the directory

WINDOWS: C:\Sample

UNIX: /home/Sample

We also need to create a jBASE H4 file called TABLEFILE, this file will hold reference to the above files. Each entry in a TABLEFILE is very similar to having an F Pointer in your "MD"

Create a directory to hold the file,

WINDOWS: C:\TSample

UNIX: /home/TSample

In the above directory create a jBASE file called TABLEFILE

d. Creating a link to the sample tables and dictionaries,

If you are using a UNIX system, <<ACCOUNT>> will be /home/sample, WINDOWS c:\Sample.

```
jed TABLEFILE TORDERS
```

```
001: <<ACCOUNT>>\ORDERS
```

```
002: <<ACCOUNT>>\ORDERS ]D
```

```
jed TABLEFILE TITEMS
```

```
001: <<ACCOUNT>>\ITEMS
002: <<ACCOUNT>>\ITEMS]D
```

```
jed TABLEFILE TCLIENTS
```

```
001: <<ACCOUNT>>\CLIENTS
002: <<ACCOUNT>>\CLIENTS]D
```

e. Adding a datasource.

You will need to let both the client and server know what type of datasource you are creating, this can be done by using one of the following :

“Attunity Configuration Manager”, This is a GUI interface installed with the windows version of Attunity, it also allows you to configure remote servers, (UNIX and WINDOWS)

“nav_util”

nav_util is a multi-functional utility supplied by Attunity, it allows jDP to setup and maintain datasources, and it can also execute queries.

To add a datasource via nav_util execute the following from the server..

```
nav_util UPD_DS TSAMPLE jDP table file=/home/TSample/TABLEFILE
```

You can view existing bindings with:-

```
nav_util view bindings
```

For more information please read the Attunity documentation.

f. Setting up the environment variables for a client...

When a client connects the Attunity driver needs to be able to see where the jBASE account is. This is done by using the following script.

```
WINDOWS: nav_login.bat
```

UNIX: site_nav_login.sh

WINDOWS:

Using jed / notepad edit %NAVROOT%\bin\ nav_login.bat

```
jed %NAVROOT%\bin\ nav_login.bat
```

```
File C:\jbase4\4.1\jdp\bin\ , Record 'nav_login.bat'           Insert
17:01:52
Command->
001 @echo off
002 SET PATH="C:\jbase4\4.1\jdp\BIN";%PATH%
003 SET NAVROOT=C:\jbase4\4.1\jdp
----- End Of Record -----
-----
```

Add the following lines to let jBASE know where things are..

```
004 SET JEDIFILEPATH=C:\Sample;.
005 SET JEDIFILENAME_MD=c:\Sample\MD
006 SET HOME=c:\Sample
```

UNIX:

```
jed $NAVROOT/bin/site_nav_login.sh
001 #
002 # Modify Attunity 3.2 environment for jBASE 4.x
003 #
...
035 # Make sure Attunity can see our demo data files
036 JEDIFILEPATH=/usr/jbc/jdp/demo
037 export JEDIFILEPATH
038
039 #
040 # Any additional site specific changes should follow
```

041

```
#####  
##
```

Add the following lines to let jBASE know where things are..

```
036 JEDIFILEPATH=/home/Sample  
037 export JEDIFILEPATH  
038 JEDIFILENAME_MD=/home/Sample  
039 export JEDIFILENAME_MD  
040 HOME=/home  
041 export HOME
```

On the client machines you also need to add a reference to the remote datasource, this can be done via the GUI or by executing the following command :

```
nav_util UPD_DS REMOTE <<datasource>>
```

EXAMPLE

```
nav_util UPD_DS REMOTE TSAMPLE
```

- g. Creating a link between the Attunity driver and the jBASE files.

If you are using a file that contains multivalued columns you will need to configure jDP so that the jDP driver can treat the multivalued columns as tables, physically they are still stored as multivalued in the jBASE files, and jDP simply creates an internal virtual table that the jDP driver can use.

This is achieved by using the “nav_util” utility,

The general syntax is as follows...

```
nav_util gen_array_table <<datasource>> <<tablename>>
```

EXAMPLE

```
nav_util gen_array_table TSAMPLE TCLIENT
nav_util gen_array_table TSAMPLE *
```

You can list all data sources with the following command :-

```
nav_util view datasources *
```

The file viewer will vary depending on whether you are running on windows or UNIX in this example we are not using multi-values so all tables should be usable via jDP without any problems.

For further information see your Attunity documentation.

Catalogue datasource

This example shows how to use a Catalogue datasource with jDP,
To create a catalogue, you must set up a catalogue directory by using the 'CreateJDPCatalog' program normally found in the jBASE Bin directory. Run the program on the server (the machine that will hold the data), which will prompt the user for three inputs: 'DSN Name' (the name to be entered into the nav.bnd file), 'Catalogue Directory' and 'Public Tables Path'.

DSN Name

This is the datasource name that the program will write into the binding file; the 'jDP' software to connect to the catalogue uses this file.

Catalogue Directory

This is the directory that the catalogue can find all its required system tables.

Public Tables Path

The directory the catalogue uses to hold the tables for the default schema.

EXAMPLE

```
jsh ~ -->CreateJDPCatalog
```

```
%JBCRELEASEDIR% is currently set to: C:\jbase4\4.1
```

```
%NAVROOT% is currently set to: C:\jbase4\4.1\jDP
```

```
Attunity version: 3.4.02.00
```

```
Enter the Data Source name or <Q>uit: CATSAMPLE
```

```
Enter absolute directory pathname in which to locate the Catalog or <Q>uit: c:\cat
```

```
Enter absolute directory pathname in which to create tables for the Public schema or <Q>uit:
```

```
c:\cat\public
```

- Creating Catalog and Public directories...
- Loading initial Catalog entries into 'c:\cat'...
- Updating the Schema file...
- Updating Object Store...
- Updating Security...

```
jDP Catalog created successfully.
```

```
jsh ~ -->
```

After the program has run, you will have to let jDP know about the new datasource.

You can do this via the windows GUI or by typing:

```
Nav_util UPD_DS CATSAMPLE jDP catalogdir=c:\cat
```

If a client machine accesses the data (i.e. the server and the client are separate machines) then the client will also need to setup a datasource entry.

```
Nav_util UPD_DS CATSAMPLE REMOTE <<remote ip>>
```

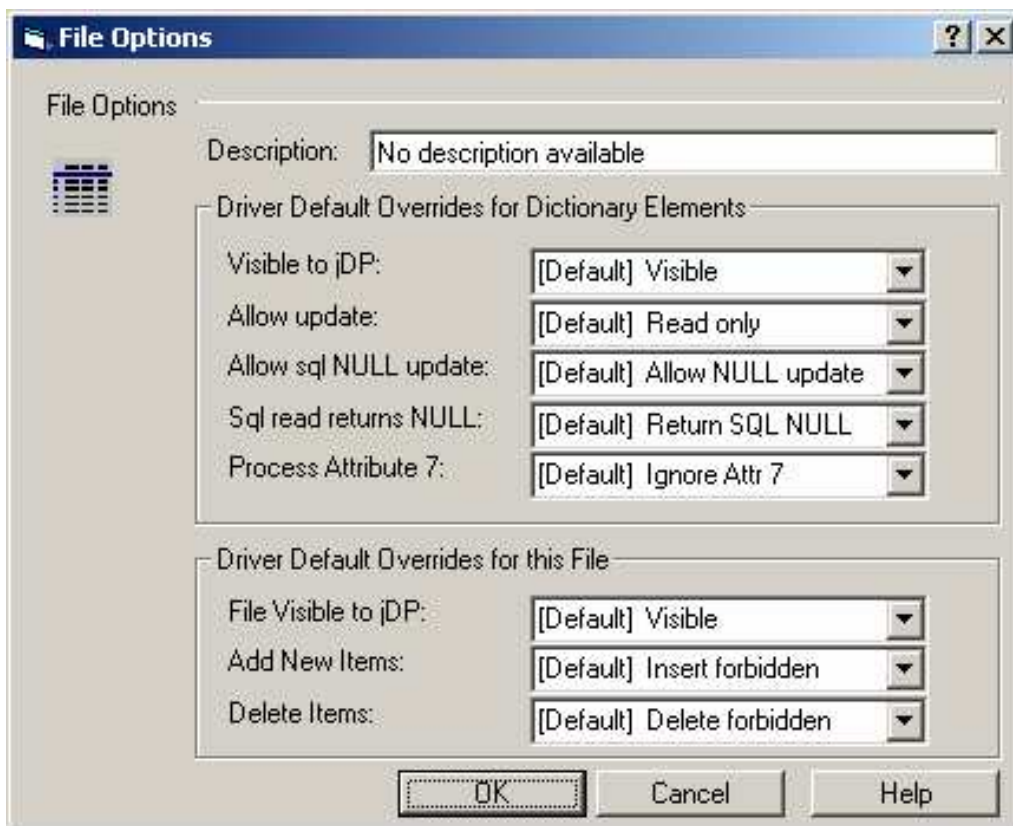
JBASE DICTIONARY CONFIGURATOR

The jBASE Dictionary Configurator (or jDC) is designed to amend standard jBASE Dictionaries to store extra data needed by jDP in order to carry out tasks in SQL, which require more information to perform operations such as joins. The jDC also allows configuration of which files are visible and updateable. By default, all files are read-only through jDP.

The jDC only operates on locally accessible files. To configure server based files, a network share mechanism needs to be set up.

Normal jBASE files contain data in one file in multi-valued format which would normally be spread amongst a number of hierarchical tables in a relational database. The jDC allows the configuration of a number of logical sub tables to allow SQL constructs to work against jBASE multi-value data in the normal manner.

To configure the settings for a file, select it and click on the properties button. A dialog should appear similar to this which allows configuration of the visibility / read-only attributes of the file.



These options are as follows;

Description	Some tools can be supplied with the string you enter here as a TABLE description.
Visible to jDP	Specifies whether the columns are hidden from jDP or not. The default, provided by the driver, is that the table is visible. Override the default here. You can override the file defaults on an individual column (dictionary entry).
Allow update	Specifies whether dictionary elements in this file are updatable by default.
Allow SQL NULL Update	If a dictionary element is updateable, this property allows or denies its update using the SQL value NULL. Note that SQL NULL is not the same as a null string "". It is roughly equivalent to "Unassigned Var", in the jBC language.
SQL read returns NULL	When reading from a column, (dictionary element), you can specify whether to return an empty element (null string "", or "" in a numeric field for instance), as the special SQL NULL or not. The driver default should only be overridden if you fully understand the concepts of SQL NULL.
Process Attribute 7	Dictionary elements that are A or S types, may have conversions/correlatives on both attribute 7 and attribute 8 of their definition. By default, anything specified on Attribute 7 is ignored. You can choose to process this attribute by default or not.
File visible to jDP	By default, the driver advertises all files that it can see. However, you can choose to remove this file from the list of advertised TABLES.
Add New Items	Inserts, Deletes and Updates are forbidden by default, for what should be obvious security concerns. If you want users to be able to add new rows (items etc), to the file, then specify this here.
Delete Items	Specify whether users are allowed to delete rows (items etc), from this file or not.

JDP AND SUBVALUES

jDP can be used to query subvalues provided the system has been setup correctly. In order to use jDP to query sub values, the environment variable JDP_AUTO_EXPAND must be set to instruct the query engine to take account of them. In addition all subvalues must be properly configured using the jDC tool such that there is a controlling multivalued for each subvalue set.