

## Table of Contents

NSC Overview.....	3
Requirements.....	3
Command Line Syntax.....	3
Starting the NSC.....	3
NSC Configuration Arguments .....	3
-host <hostname or IP address> .....	3
-port <port#>.....	3
-debug .....	4
NSC Commands.....	4
Command.....	4
-cmd .....	4
Batch Command.....	4
batch -file <file> [-repetitions <n>] [-sleep <msec>] [-gc] [-log <log-file>] [-qa] .....	4
Arguments for -cmd batch .....	4
Table Commands .....	5
tbllist .....	5
tblcreate -table <name> -file <def-file> .....	5
tblload -table <name> -file <csv-file> [-settableinfo <text>] [<csv-options>] .....	6
csv-options .....	7
fastload -table <name> -file <csv-file> [-create -replacenotadd] <csv-options> .....	7
tbldelta -table <name> -file <delta-file> -fieldkeynumber <field-number> .....	8
tbldelete -table <name> .....	8
tblmove -oldtable <name> -newtable <new-name> .....	8
tbldump -table <name> [-file <dump-output-file>] .....	8
tbllock -table <name> .....	8
tblunlock -table <name> .....	8
Table Statistics Command:.....	9
tblstats -table <name> .....	9
Thesaurus Commands:.....	9

thlist .....	9
thcreate -thesaurus <name> -file <file> .....	9
wdcreate -thesaurus <name> -file <file> .....	9
cthcreate -thesaurus <name> -file <file> .....	10
thdelete [-thesauruslist <name1,name2,...>] .....	10
Record Commands: .....	10
recget -table <name> -keys <key1,key2,...> .....	10
recdelete -table <name> -keys <key1,key2,...> [-domaxwork] .....	10
recadd -table <name> -file <file> [-estnumrecords <n>] <csv-options> .....	10
recreplace -table <name> -file <file> <csv-options> .....	10
Character Map Commands: .....	10
maplist .....	10
mapcreate -mapname <name> [-mapchars <map-file>] [-foldcase] .....	11
[-foldldiacritics] [-punctuation <c>] [-whitespace <c>] .....	11
Index Command: .....	11
idxlist -tables <table1,table2,...> .....	11
Query Commands: .....	11
search -query <query-str> -table <name> [-fields <field1,field2,...>] .....	11
querygen .....	11
Model Commands: .....	12
rlcreate -modelname <name> -file <model-file> .....	12
rldelete -modellist <model1,model2,...> .....	12
rllist .....	12
Checkpoint Commands: .....	13
getcheckpointstatus -table <name> .....	13
checkpoint -tables <table1,table2,...> .....	13
restore -tables <table1,table2,...> .....	13
Server Interface Control Commands: .....	13
debuggingon .....	13
debuggingoff .....	13
getconnectionpoolingpolicy .....	13
setconnectionpoolingpolicy -policy <off local common> .....	13

version.....	13
Server Status and Control Commands: .....	14
isdecisionengine.....	14
isgipenbled .....	14
issortenabled.....	14
svrshutdown [-pidfile <file-path>] [-waittime <wait-time >].....	14
svrnoop .....	14
svrversion.....	14
svrlogon.....	15
svrlogoff .....	15

## NSC Overview

The NSC is a command line utility used to interact with the TIBCO® Patterns server. Commands can be issued one at a time or from a batch file. The NSC can be used to create, update, remove, view objects on the matching engine or perform searches. NSC serves as both a potentially useful tool in its own right and as an example of the use of the Java API to the TIBCO® Patterns server.

## Requirements

The NSC command line utility was built to be compatible with Java version 1.5 or later.

## Command Line Syntax

All commands and arguments will be prefaced with a '-'. Arguments that are optional are in square brackets []. Parameters to arguments are shown in angle brackets <>. All parameters are required.

## Starting the NSC

How to issue commands from the NSC:

From the OS command line:

```
java -jar nsc.jar [-host hostname] [-port port#] [-debug] -cmd <cmd...>
```

## NSC Configuration Arguments

The configuration arguments are optional. Defaults will be used if not specified. The configuration arguments are only valid on the initial execution of NSC. Issuing -host or -port on a secondary line in a batch file will have no effect since the secondary commands are already inside the initial connection.

### **-host <hostname or IP address>**

The host parameter specifies the host location where the TIBCO® Patterns server is installed. The default value for '-host' is "localhost" i.e. the current machine. You may specify the name or IP address of another machine if the host resides on a remote machine. NSC cannot be used to communicate with a TIBCO Patterns server that does not have IPv4 enabled for input.

### **-port <port#>**

The '-port' parameter specifies the port to establish a connection on when communicating with the TIBCO® Patterns server. The default value is "5051". You may specify a different port if the default is not being used.

## **-debug**

The debug option turns on a debug trace of communications between NSC and the TIBCO® Patterns server . The debug trace is output to the stderr and is quite lengthy. Debug tracing is off by default.

## **NSC Commands**

### **Command**

#### **-cmd**

All NSC commands are prefaced by the ‘-cmd’ parameter. Command line switches that follow the specified cmd parameter are arguments to the command.

### **Batch Command**

The NSC is capable of accepting a series of commands. The commands should reside in a file. Each command in the file will start on its own line. Blank lines are not permitted. Processing will stop if a blank line is encountered. A line will be considered a comment if it starts with ‘//’.

**batch -file <file> [-repetitions <n>] [-sleep <msec>] [-gc] [-log <log-file>] [-qa]**

#### **Example Batch File:**

```
// Create a table with indexes.  
-cmd tblcreate -table mytable -file "C:\Netrics\Data\CreateNewTable\tabdef.csv"  
// Display the indexes on the table.  
-cmd idxlist -tables mytable
```

### **Arguments for -cmd batch**

When using the batch command, the following arguments are valid:

#### **-repetitions <n>**

Repetitions will cause the entire batch file to be rerun ‘n’ number of times. The default is ‘1’ which will cause the batch file to be issued only 1 time.

#### **-sleep <msec>**

Sleep will cause there to be a pause in between ‘-cmd’ executions. The ‘-sleep’ is specified in milliseconds (i.e. 1000 = 1 second). The default value is ‘0’ which will not delay processing.

#### **-gc**

The ‘-gc’ argument will force Java garbage collection immediately after the a ‘-cmd’ has completed. The default is to let normal garbage collection occur.

### ***-log <log-file>***

The ‘-log’ argument will direct output to the specified file during the batch file execution. The default is for output to go to the stdout and stderr.

### ***-qa***

The ‘-qa’ argument suppresses batch processing informational content such as timestamps.

## **Table Commands**

The following are commands used for interacting with the data tables on the matching engine.

### **tbllist**

The ‘tbllist’ command will return a list of all tables loaded in the matching engine.

Example output:

```
List: GENERIC
  DBDESCRIPTOR: names
  CHECKPOINT_STATUS: Never Checkpointed
  DBNUMFIELDS: 8
  FIELDNAMES: ["last","first","ssn","street","city","state","zip","id"]
  FIELDTYPES: [ 5, 5, 5, 5, 5, 5, 5, 5]
  CHARMAPS: ["=STD=","=STD=","=STD=","=STD=","=STD=","=STD=","=STD=","=STD="]
  DBGIPFILTER: true
  DBSORTFILTER: false
  DBNUMRECORDS: 10012
  DBKEYTREEKBYTES: 357
  DBRECFIELDKBYTES: 661
  DBHEADERKBYTES: 223
  DBIDXTOTALKBYTES: 5616
  DBTOTALKBYTES: 6858
```

### **tblcreate -table <name> -file <def-file>**

The ‘tblcreate’ command is used to create a table. It allows the user to define:

- Field names.
- Field types.
- Field maps.
- Field indexes.

The ‘tblcreate’ command has two arguments: *name* the name of the table and *<def-file>* the file name of a table definition file.

The table definition file contains one line for each field in the table with each line having the comma separated values as shown below:

Fieldname,Field type,Field map[, index file][,index file]

#### *Example of a Table Definition File:*

```
col1,5,=STD=,C:\Netrics\Data\addPIndex\state.txt,C:\Netrics\Data\addPIndex\state2.txt
col2,5,=STD=
col3,5,=STD=
col4,5,=STD=,C:\Netrics\Data\addPIndex\state3.txt
col5,5,=STD=
col6,5,=STD=
```

Each row defines a different field and fieldname, field type and a field map must be specified.

#### About Field Names

Fieldnames must not contain spaces or special characters<sup>1</sup>.

#### About Field Types

Field Types determine how the matching engine classifies the field. Field types are specified as:

FLDTYP_DATE	10
FLDTYP_DATETIME	12
FLDTYP_FLOAT	8
FLDTYP_INT	6
FLDTYP_SRCHTEXT	5
FLDTYP_TEXT	4

Only field type 5 is searchable using fuzzy search by the matching engine.

#### About maps

In the event that there is no custom map to be assigned, the default map '=STD=', should be specified. If a custom map is to be specified, it must be loaded prior to the 'tblcreate' to be available for the table creation (see the '-cmd mapcreate' for more details on creating a map).

#### About Indexes

Indexes are used to speed predicate evaluation. See your TIBCO representative for more details on the creation of indexes.

**tblload -table <name> -file <csv-file> [-settableinfo <text>] [<csv-options>]**

The tblload command creates a table and loads a csv file into the matching engine. Options:

**name** is the name of the table to be created. This is required

---

<sup>1</sup> Note this is a restriction of NSC, the server supports blanks and some special characters. See the Concepts Guide for further details on valid field names.

**csv-file** is the name of a CSV file containing the records to be loaded. The first line **must** be a header line defining the names of the fields of the file. This is required.

**text** is optional arbitrary text associated with this table.

For the **tblload** command **-fieldnamesfirst** in the **csv-options** must be specified.

## csv-options

**csv-options** is a collection of arguments used to define the format of the CSV file containing the records. These options are common to a number of commands. The options are:

<b>-encoding &lt;char-set&gt;</b>	the character set encoding of the file. Valid values are “ <b>latin1</b> ” and “ <b>UTF-8</b> ”. The default is “ <b>latin1</b> ”.
<b>-domaxwork</b>	if given records in the CSV file with formatting errors or duplicate keys are quietly ignored.
<b>-leadingkey</b>	if given the first field in each data record is assumed to be the key field for the record. This field is unnamed (doesn’t appear in any header record of field names).
<b>-initialkey &lt;n&gt;</b>	if keys are being generated automatically the key sequence starts with <b>n</b> . Default is zero.
<b>-fieldnamesfirst</b>	if given the first line of the CSV file is a header line defining the names of each field. Default is no field name header line.
<b>-fieldtypesfirst</b>	if given the first line of the CSV file (after any field name header line) defines the field types for each field. The field types are integer values as listed in “About Field Types” above. Default is no field types header line.
<b>-isparent</b>	if given the table is created as a parent table. Default is the table is not a parent table.
<b>-parenttable &lt;table-name&gt;</b>	if given this is a child table. The <i>table-name</i> given must be the name of an existing parent table. It is an error to provide this option if <b>-isparent</b> is given. Default is the table is not a child table.
<b>-keyfieldindex &lt;n&gt;</b>	the index (zero based) of the field that is to be used as the record key. This field is not loaded as a data field.
<b>-keyfieldname &lt;name&gt;</b>	the name of the field that is to be used as the record key. This field is not loaded as a data field.
<b>-parentkeyfield &lt;name&gt;</b>	the name of the field that is used as the parent key of a child table. This field is not loaded as a data field. This is required for child tables, that is, if the <b>-parenttable</b> option is given. It is not allowed for non-child tables.

Only one of **-leadingkey**, **-keyfieldindex** or **-keyfieldname** may be specified. If none are specified keys are generated automatically. The **-initialkey** argument is only used if keys are being generated automatically.

## fastload -table <name> -file <csv-file> [-create|-replacenotadd] <csv-options>

Use the fastload command to significantly decrease table load time. Options:

**name** is the name of the table to be created. This is required.



**csv-file** is the name of a CSV file containing records to be loaded. This file must be accessible by the Matching Engine. This is required.

**-create** if present a new empty table is created. Any existing table with the same name is deleted. The default is to assume the table already exists.

**-replacenotadd** if this is specified records in an existing table are replaced with those from the CSV file instead of being added as new records. The CSV file must contain record keys.

**<csv-options>** CSV file format options as defined for **tblload**.

### **tbldelta -table <name> -file <delta-file> -fieldkeynumber <field-number>**

The **tbldelta** command will process a delta file against an existing table. The delta file, currently only CSV is supported, must have a 1 character column in the first position that contains (i/u/d) i=insert, u=update, d=delete. The delta file must contain the key value to identify the action record which is identified by the **-fieldkeynumber** argument. The **-fieldkeynumber** value is zero based.

### **tbldelete -table <name>**

Use the **tbldelete** command to delete an existing table from the matching engine.

### **tblmove -oldtable <name> -newtable <new-name>**

Use the **tblmove** command to rename a table. This command is typically used to instantly replace an existing table with a freshly loaded version. Pre-load a table that is to be updated with a temporary name and then 'move' it to the existing table. This update/replace methodology insures that there is no down time for the active table. Queries being run against the **-newtable** will be queued until the **tblmove** operation completes.

### **tbldump -table <name> [-file <dump-output-file>]**

The **tbldump** command will stream the entire contents of a table to the console (not recommended for large tables above 10K records) or to a file if the **-file** option is given. The output varies depending on what the destination is. Output to the console will be prefaced with a 'key=' identifier. Output to a file will not contain a key identifier and will mirror the layout in the table. The original table will not be affected.

### **tbllock -table <name>**

Use the **tbllock** command to place a lock on a table. Using a lock should be done with the knowledge that queries against a table will not be queued while the table is locked and will fail. Any application making use of the **tbllock** command should be prepared to handle the resulting failure response from the query attempt. Table updates do not require that the table be locked and use of this command is discouraged.

### **tblunlock -table <name>**

Use the **tblunlock** command to unlock a previously locked table.

## Table Statistics Command:

### **tblstats -table <name>**

The tblstats command returns various statistical information about the specified table.

Example of output against a table called 'names'.

```
-- Table Statistics: names
The status of any checkpoint for this table: Never Checkpointed
The total size of the overhead for a table (in kbytes): 223
The total size of the indexing information for the table (in kbytes): 5616
The total size of the tree structure which holds the record keys for the
table (in kbytes): 357
The number of records currently in the table: 10012
The total memory used by the process (in kbytes. Only available on Linux.): -
1
The total size of the overhead for storing the field structure of the table
(in kbytes): 661
The table info field for this table. It will be null if no table info was
specified at load time:
The total size of the table, all structures included (in kbytes): 6858
Whether or not the GIP filter is turned on for the database: true
-- Field Level Information
Field Name: last : Field Type: 5 : Searchable: true
Field Name: first : Field Type: 5 : Searchable: true
Field Name: ssn : Field Type: 5 : Searchable: true
Field Name: street : Field Type: 5 : Searchable: true
Field Name: city : Field Type: 5 : Searchable: true
Field Name: state : Field Type: 5 : Searchable: true
Field Name: zip : Field Type: 5 : Searchable: true
Field Name: id : Field Type: 5 : Searchable: true
```

## Thesaurus Commands:

### **thlist**

The thlist command returns a list of all the thesauri loaded in the matching engine.

### **thcreate -thesaurus <name> -file <file>**

The thcreate command is used to create a thesaurus in the matching engine.

**file** is the name of a standard thesaurus definition file. This must be accessible by the Matching Engine. See the TIBCO® Patterns server documentation for the format of this file.

### **wdcreate -thesaurus <name> -file <file>**

The wdcreate command is used to create a weighted dictionary in the matching engine.

**file** is the name of a weighted dictionary definition file. This must be accessible by the Matching Engine. See the TIBCO® Patterns server documentation for the format of this file.

### **cthcreate -thesaurus <name> -file <file>**

The cthcreate command is used to load a combined thesaurus/weighted dictionary in the matching engine.

**file** is the name of a weighted dictionary definition file. This must be accessible by the Matching Engine. See the TIBCO® Patterns server documentation for the format of this file.

### **thdelete [-thesauruslist <name1,name2,...>]**

The thdelete command is used to delete a thesaurus, weighted dictionary or combined thesaurus/dictionary that is currently loaded in the matching engine. Warning: Specifying thesauri is optional. Failure to specify a thesaurus will delete all thesauri.

## **Record Commands:**

### **recget -table <name> -keys <key1,key2,...>**

The recget command will retrieve each of the records with the specified key(s) and display them to the standard out.

### **recdelete -table <name> -keys <key1,key2,...> [-domaxwork]**

The **recdelete** command will delete each of the records with the specified keys. If the **-domaxwork** option is specified non-existent records will be quietly ignored, otherwise if any of the specified keys are not in the table the entire command will fail and no records will be deleted.

### **recadd -table <name> -file <file> [-estnumrecords <n>] <csv-options>**

The recadd command will add records to a table from the specified file. The **estnumrecords** option is used to give the server a hint as to the size of the load so that it can choose the optimal method of loading. **csv-options** is as defined for the **tblload** command.

### **recreplace -table <name> -file <file> <csv-options>**

The recreate command will replace records in a table from a specified file. **csv-options** is as defined for the **tblload** command. A key field must be specified, auto-generated keys are not allowed.

## **Character Map Commands:**

### **maplist**

The maplist command will return a list of all the character maps loaded in the matching engine.

**mapcreate -mapname <name> [-mapchars <map-file>] [-foldcase] [-folddiacritics] [-punctuation <c>] [-whitespace <c>]**

The mapcreate command will create a character map in the matching engine. Once created the character map can be used in **tblload** commands as the character map applied to a field. Options are:

<b>-mapname &lt;name&gt;</b>	the unique identifying name for this character map.
<b>-mapchars &lt;map-file&gt;</b>	used to set explicit character mappings. <i>Map-file</i> is the name of a file that consists of sets of 3 characters: <b>&lt;from&gt;&lt;to&gt;&lt;new-line&gt;</b> specifying that character <b>from</b> is mapped to character <b>to</b> before comparing strings. The <b>new-line</b> character is used to visually separate the map character sets. So the file consists of a set of lines consisting of exactly two characters. (Except that either <b>from</b> or <b>to</b> may be the new-line character.)
<b>-foldcase</b>	if this is present all letters are mapped to a common letter case.
<b>-folddiacritics</b>	if present diacritic marks are stripped from letters.
<b>-punctuation</b>	if present all punctuation characters are mapped to the character <b>c</b> .
<b>-whitespace</b>	if present all whitespace characters are mapped to the character <b>c</b> .

## Index Command:

**idxlist -tables <table1,table2,...>**

Use the idxlist command to list all the indexes associated with the specified table(s).

## Query Commands:

**search -query <query-str> -table <name> [-fields <field1,field2,...>]**

The **search** command allows you to run a simple query against an existing table.

<b>query-str</b>	is the string to be matched.
<b>name</b>	is the name of the table to be searched.
<b>Field1,field2,...</b>	is a comma separated list of the fields with the table to be searched. All of the named fields must be searchable text fields in the table. If this is not given all searchable text fields in the table are searched.

## querygen

The querygen command is a search query generation tool that creates a sample Java class file. The class file produced will contain 3 different query types; "Simple", "And", and "Cognate". The class file is fully runnable after being compiled, but in most cases, modification will be desired and in some cases where input columns do not match with the table in the matching engine, required. Once the class file is

created you can run the class file as a standalone program or if you want to feed the compiled executable jar file queries from the command line, use the “-cli” argument. Within the generated class file you may specify the output to create a HL page with color coded output or a CSV file. To specify the output format, within the generated class file, change the output file extension as documented in the generated java file. When issuing the querygen command, the output file location should be specified using double backslashes, “\\”.

The quergen command can also be used to run batch matches by using the “-batchfile” argument. If the columns in the batch input file do not match, some manipulation of the variables “data[n]” will be required in the generated Java class file.

If using the -cli option, you will need the Apache Commons CLI Java jar files to compile.

```
-cmd querygen -table <table> [-name <name>] [-file <file(use \\>)] [-cli] [-batchfile <input file>]
```

Examples of NSC querygen calls:

Regular:

```
-cmd querygen -table names -file c:\\dev\\ -name java_sample
```

CLI:

```
-cmd querygen -table names -file c:\\dev\\ -cli
```

Batch File:

```
-cmd querygen -table names -batchfile c:\\data\\names.csv -file c:\\dev\\
```

## Model Commands:

**rlcreate -modelname <name> -file <model-file>**

The **rlcreate** command loads a TIBCO machine learning model contained in the file **model-file** into the Matching Engine. The model is named **name**.

**rldelete -modellist <model1,model2,...>**

The **rllist** returns a list of all the models loaded in the engine.

**rllist**

The **rllist** returns a list of all the models loaded in the engine.

## Checkpoint Commands:

### **getcheckpointstatus -table <name>**

Use the **getcheckpointstatus** command to return the checkpoint status of the named file. If the table has been checkpointed the time of the last checkpoint is given.

### **checkpoint -tables <table1,table2,...>**

Use the **checkpoint** command to write a snapshot of the specified table(s) to disk.

### **restore -tables <table1,table2,...>**

Use the **restore** command to load specified table(s) from snapshots created by a previous **checkpoint** command.

## Server Interface Control Commands:

### **debuggingon**

Use the **debuggingon** command in batch commands to turn debug tracing of all interactions with the TIBCO® Patterns server .

### **debuggingoff**

Use the **debuggingoff** command in batch commands to turn debug tracing off.

### **getconnectionpoolingpolicy**

Use the **getconnectionpoolingpolicy** command to determine what level of connection pooling is enabled. 0 = No connection pooling. 1 = Matching engine has own connection pooling. 2 = Shared connection pooling. The default is "2".

### **setconnectionpoolingpolicy -policy <off|local|common>**

Use the **setconnectionpoolingpolicy** command to set the connection pooling policy. **-policy** must be set to one of:

- off** do not use connection pooling when communicating with the matching engine. Every command is a new connection.
- local** each instance of NetricsServerInterface has a separate pool of active connections to the matching engine.
- common** there is a single common pool of connections for all NetricsServerInterface objects.

### **version**

The version command returns the release level of the of the Java interface code that is being used.

## Server Status and Control Commands:

### isdecisionengine

Use the **isdecisionengine** command to determine if the server is a TIBCO Matching Platform server with the machine learning features.

### isgipenabled

Use the **isgipenabled** command to determine if the GIP prefilter for performance optimization is currently enabled on the TIBCO® Patterns server.

### issortenabled

Use the **issortenabled** command to determine if SORT prefilter for performance optimization is currently enabled on the TIBCO® Patterns server.

### svrshutdown [-pidfile <file-path>] [-waittime <wait-time >]

This command will cause the TIBCO® Patterns server to perform a controlled shutdown.

If the **-pidfile** option is given it checks for the existence of the file: **<file-path>**, if the file doesn't exist, the command completes successfully without further action. If the file exists it sends a shutdown command to the TIBCO® Patterns server and then waits **<wait-time >** seconds (default 10 seconds) for the pid file **<file-path>** to be removed, checking once a second. If the file is removed it exits successfully. If it is not removed after **<wait-time>** seconds the file is read to retrieve the process ID and that process is killed. If the kill is successful the command completes successfully, otherwise it completes with an error message. This sequence is designed to ensure the TIBCO® Patterns server is shut down before this command returns.

If the **-pidfile** option is not given it sends a shutdown command to the TIBCO® Patterns server and completes successfully.

### svrnoop

This command performs no action. It is used to verify that a connection to the server can be made and that the server is up and operational.

### svrversion

The **svrversion** command returns the release level of the TIBCO® Patterns server.

## svrlogon

The **svrlogon** command turns on matching engine query logging. Note that the matching engine query logging functionality must be enabled at engine startup by specifying a query log file or this command will fail.

## svrlogoff

The **svrlogoff** command turns off matching engine query logging. Note that the matching engine query logging functionality must be enabled at engine startup by specifying a query log file or this command will fail.

