TIBCO® Patterns

Utilities

Version 6.1.2 | June 2024

Contents

Patterns Data Transfer Tool - Overview

This section describes the process of using Patterns Data transfer tool.

TIBCO® Patterns stores multiple types of data while performing its role as an inexact matching engine. Transferring this data between Patterns installations can be difficult, particularly if the installation architectures are incompatible.
For example, transferring data from a single-server installation to a clustered installation.

The Patterns Data Transfer Tool (PDTT) is a Java™-based command-line utility, which transfers data between installations with a single command, regardless of installation architectures.

Limitations

1. PDTT has limited support for TIBCO Patterns versions prior to 6.1.2:
    - The source server and target server must be at 5.6 version or higher.
    - When the source server is 5.6 or 5.7 version, only tables can be transferred.
2. Transferring from a higher version server to a lower version server is not supported.

    PDTT does not block such attempts, but it is not a supported use case and is likely to fail.
3. PDTT should be run in isolation. No other applications must be modifying data on the source or target server while PDTT is running. Applications can read data but not write it.
4. PDTT only performs full transfers and attempts to transfer all objects.
5. Transfer of an object fails if the object exists on the target server.

    For character-maps, thesauri, and RLink models behavior can be changed with the `-i` command-line option. The `-i` option enables over-writing existing thesauri and RLink-models. Tables and character-maps cannot be overwritten, even with the `-i` option.
6. One transaction is used per object.

    If PDTT fails, the following actions occur:
    - Objects that were completely transferred remain on the target server.
    - Objects that were not completely transferred are removed from the target server.
7. Parallel transfer

    Tables can be large. To minimize the transfer time and memory usage, PDTT transfers

tables in parallel. All other objects are transferred one at a time.

8. Resource usage
    - Memory: PDTT typically uses less than 2 GB of memory. Transferring large RLink Learn models requires up to an additional 6 GB of memory.
    - CPU: PDTT uses one thread per table. Transfer of other objects is single-threaded and takes place before transferring tables.
9. Logging

    PDTT is compatible with Apache log4j2. This provides for setting the location of the log output, log levels, and so on. The default log level is *info*.

Usage

As Patterns Data Transfer Tool is a command-line based tool, the following command is used for data transfer:

```
java [jvm_options] -jar  TIB_tps_datatransfer.jar  source_server  target_server
[-i] [-o]
```

Command	Description
`jvm_options`	Any applicable options to your Java Virtual Machine
`source_server`	Patterns installation that data is transferred from
`target_server`	Patterns installation that data is transferred to
`-i`	Objects that exist on the target server do not cause an error
`-o`	Override server-version validations.

Both, *source_server* and *target_server* use the following format:

```
version:hostname:port[:SSL]
```

Option	Description
version	Version of the Patterns server. Example: 6.1.2
hostname	DNS-name or IP-address where the Patterns server is running
port	The TCP port that the Patterns server is listens on
SSL	Use SSL-encrypted communications

The versions specified on the command line are checked against the versions returned by the servers. If either does not match, an error is logged and the transfer is not attempted. You can override this behavior with the -o option. If -o is specified and the versions do not match, a warning is logged and the transfer is attempted.

Example:

```
java –jar TIB_tps_datatransfer.jar 5.6:localhost:8051:SSL  6.1:otherhost:5055
```

This command is used to commence transfer from a TIBCO® Patterns 5.6 server to an TIBCO® Patterns 6.1.0 version. The 5.6 Patterns server uses SSL-encrypted communications on port 8051 on the local host. The 6.1.2 Patterns server uses plain-text communications on port 5055 on machine `otherhost`. Since *-i* is not specified, any object that already exists on *otherhost* causes an error.

A Sample Run of Patterns Data Transfer Tool

Note: The z440 machine is associated with the 5051 port and localhost is associated with the 5055 port. It is important to maintain consistency with these associations throughout this section when executing commands.

This example uses PDTT to transfer data from a Patterns 5.6.0 instance on port 5051, on the `z440` machine to a Patterns 6.1.0 instance on the local machine on port 5055.

Using the NSC command-line tool, we can see the tables on the source instance:

```
java –jar NSC.jar –host z440 –port 5051 –cmd tbllist
```

```
>>> nsc start.
```

```
List: GENERIC
DBDESCRIPTOR: join--names
CHECKPOINT_STATUS: Last Checkpoint: 2022/05/11-10:31:33
TRAN_ID: 0
TRAN_OBJ_STATE: 1
DBINFO:
DBNUMFIELDS: 6
FIELDNAMES: ["last","first","ssn","street","city","zip"]
FIELDTYPES: [5, 5, 5, 5, 5, 5]
FIELDBYTELCOUNTS: [337, 281, 486, 930, 506, 270]
FIELDCHARLCOUNTS: [337, 281, 486, 930, 506, 270]
CHARMAPS: ["=STD=","=STD=","=STD=","=STD=","=STD=","=STD="]
DBGIPFILTER: true
DBGIPGPU: false
GIPGRAMLEN: 3
DBSORTFILTER: false
DBPSIFILTER: false
DBNUMRECORDS: 54
DBKEYTREEKBYTES: 12
DBRECFIELDKBYTES: 3
DBHEADERKBYTES: 130
DBIDXTOTALKBYTES: 644
DBTOTALKBYTES: 1814
PROCTOTALKBYTES: 334076
TABLE_TYPE: Child
PARENT_TBL: join--states
```

```
List: GENERIC
DBDESCRIPTOR: join--states
CHECKPOINT_STATUS: Last Checkpoint: 2022/05/11-10:31:33
TRAN_ID: 0
TRAN_OBJ_STATE: 1
DBINFO:
DBNUMFIELDS: 4
FIELDNAMES: ["codeIdx","name","flower","latin-flower"]
FIELDTYPES: [5, 5, 5, 5]
FIELDBYTELCOUNTS: [100, 422, 647, 800]
FIELDCHARLCOUNTS: [100, 422, 647, 800]
CHARMAPS: ["=STD=","=STD=","=STD=","=STD="]
DBGIPFILTER: true
DBGIPGPU: false
GIPGRAMLEN: 3
DBSORTFILTER: false
DBPSIFILTER: false
DBNUMRECORDS: 50
```

```
DBKEYTREEKBYTES: 12
DBRECFIELDKBYTES: 2
DBHEADERKBYTES: 130
DBIDXTOTALKBYTES: 3833
DBTOTALKBYTES: 5001
PROCTOTALKBYTES: 334076
TABLE_TYPE: Parent
CHILD_TBLS: ["join--zips","join--names"]
```

```
List: GENERIC
DBDESCRIPTOR: join--zips
CHECKPOINT_STATUS: Last Checkpoint: 2022/05/11-10:31:33
TRAN_ID: 0
TRAN_OBJ_STATE: 1
DBINFO:
DBNUMFIELDS: 4
FIELDNAMES: ["ZipCodeType","City","StateIdx","Location"]
FIELDTYPES: [5, 5, 5, 5]
FIELDBYTELCOUNTS: [438, 531, 106, 1020]
FIELDCHARLCOUNTS: [438, 531, 106, 1020]
CHARMAPS: ["=STD=","=STD=","=STD=","=STD="]
DBGIPFILTER: true
DBGIPGPU: false
GIPGRAMLEN: 3
DBSORTFILTER: false
DBPSIFILTER: false
DBNUMRECORDS: 56
DBKEYTREEKBYTES: 12
DBRECFIELDKBYTES: 2
DBHEADERKBYTES: 130
DBIDXTOTALKBYTES: 1201
DBTOTALKBYTES: 2370
PROCTOTALKBYTES: 334076
TABLE_TYPE: Child
PARENT_TBL: join_states
>>> nsc end.
```

Enter the following command to run the transfer:

```
java -jar TIB_tps_datatransfer.jar 5.6:z440:5051 6.1:localhost:5055
```

After the transfer, the following command is used to view the tables on the target instance:

```
java -jar NSC.jar -host localhost -port 5055 -cmd tbllist
```

```
//Tables on target server//
>>> nsc start.
List: GENERIC
DBDESCRIPTOR: join--names
OBJ_ID: 56485e82-8513-4a14-9b8f-9c79184bb62c
CHECKPOINT_STATUS: Never Checkpointed
TRAN_ID: 0
TRAN_OBJ_STATE: 1
DBINFO:
DBNUMFIELDS: 6
FIELDNAMES: ["last","first","ssn","street","city","zip"]
FIELDTYPES: [5, 5, 5, 5, 5, 5]
FIELDBYTELCOUNTS: [337, 281, 486, 930, 506, 270]
FIELDCHARLCOUNTS: [337, 281, 486, 930, 506, 270]
CHARMAPS: ["=STD=","=STD=","=STD=","=STD=","=STD=","=STD="]
DBGIPFILTER: true
DBGIPGPU: false
GIPGRAMLEN: 3
DBSORTFILTER: false
DBPSIFILTER: false
DBNUMRECORDS: 54
DBKEYTREEKBYTES: 12
DBRECFIELDKBYTES: 3
DBHEADERKBYTES: 130
DBIDXTOTALKBYTES: 644
DBTOTALKBYTES: 1816
TABLE_TYPE: Child
PARENT_TBL: join--states
```

```
List: GENERIC
DBDESCRIPTOR: join--states
OBJ_ID: 1b95de52-4dd1-4863-ba6f-fe2e02a7b80f
CHECKPOINT_STATUS: Never Checkpointed
TRAN_ID: 0
TRAN_OBJ_STATE: 1
DBINFO:
DBNUMFIELDS: 4
FIELDNAMES: ["codeIdx","name","flower","latin-flower"]
FIELDTYPES: [5, 5, 5, 5]
FIELDBYTELCOUNTS: [100, 422, 647, 800]
FIELDCHARLCOUNTS: [100, 422, 647, 800]
CHARMAPS: ["=STD=","=STD=","=STD=","=STD="]
DBGIPFILTER: true
DBGIPGPU: false
GIPGRAMLEN: 3
DBSORTFILTER: false
```

```
DBPSIFILTER: false
DBNUMRECORDS: 50
DBKEYTREEKBYTES: 12
DBRECFIELDKBYTES: 2
DBHEADERKBYTES: 130
DBIDXTOTALKBYTES: 3833
DBTOTALKBYTES: 5003
TABLE_TYPE: Parent
CHILD_TBLS: ["join--names","join--zips"]
```

```
List: GENERIC
DBDESCRIPTOR: join--zips
OBJ_ID: 88d2b00b-faf6-4cbd-9506-b3d2e8abc697
CHECKPOINT_STATUS: Never Checkpointed
TRAN_ID: 0
TRAN_OBJ_STATE: 1
DBINFO:
DBNUMFIELDS: 4
FIELDNAMES: ["ZipCodeType","City","StateIdx","Location"]
FIELDTYPES: [5, 5, 5, 5]
FIELDBYTELCOUNTS: [414, 515, 106, 996]
FIELDCHARLCOUNTS: [414, 515, 106, 996]
CHARMAPS: ["=STD=","=STD=","=STD=","=STD="]
DBGIPFILTER: true
DBGIPGPU: false
GIPGRAMLEN: 3
DBSORTFILTER: false
DBPSIFILTER: false
DBNUMRECORDS: 54
DBKEYTREEKBYTES: 12
DBRECFIELDKBYTES: 2
DBHEADERKBYTES: 130
DBIDXTOTALKBYTES: 1201
DBTOTALKBYTES: 2372
TABLE_TYPE: Child
PARENT_TBL: join--states
>>> nsc end.
```

Patterns Data Synchronizer - Overview

TIBCO® Patterns stores data tables while performing its role as an inexact matching engine. For some installations, these tables must be synchronized with tables stored in another database system. The Patterns Data Synchronizer (PDS) provides a basic solution for performing synchronization to Patterns from another data source.

PDS is a Java™-based command line utility. PDS obtains changes by polling the source database. PDS processes the changes in batches. Each PDS command synchronizes a single table. For multiple tables, run one PDS command per table.

This section describes the process of using Patterns Data Synchronizer with TIBCO® Patterns.

Terminology

Term	Description
Source database	A database containing a table that PDS will synchronize to a Patterns server.
JDBC	Java™ Database Connectivity. It is a standard part of Java™.
YAML	A human-readable data language.

Prerequisites

Before using Patterns Data Synchronizer, familiarize yourself with the following information:

- Editing YAML files.
- Apache™ JDBC, which PDS uses to communicate with the source database. Before you begin, see the database vendor's documentation for the following details:

  - Formatting a JDBC connection string. Place the connection string in the PDS configuration file.

  - The required driver jar files. Place these in the Java™ class-path.

- Application Database Maintenance: PDS accesses a change-log table and a state table in the source database. Your application must create these two tables in the source database. It must also insert a row into the change-log table each time a record is changed.

YAML
Configuring PDS requires familiarity with editing YAML files.

JDBC
PDS uses JDBC to communicate with the source database.

- Consult your database vendor’s documentation on formatting a JDBC connection string. Place the connection string in the PDS configuration file.
- PDS packages publically-available JDBC drivers for Postgres™ and Microsoft SQL Server™.
- For other databases, consult your vendor's documentation for require jar files and the driver class-name. Place the driver jar file(s) in the Java™ class-path. Configure the driver class-name in the yaml configuration file using the jdbcDriver setting.
- Consult your database vendor’s documentation for required *jar* files. Place these in the Java™ class-path.

Application Database Maintenance
PDS accesses a *change-log* table and a *state* table in the source database. Your application must create these two tables in the source database. It must also insert a row into the *change-log* table each time a record is changed.

Source Database: *Change-log* and *State* Tables

PDS accesses a *change-log* table and a *state* table in the database. Your application must create these in the source database.

Change-log Table

The change-log table can have any table name that is valid in the source database. This name is entered in the PDS configuration file.

Your application must populate changes into the *change-log* table. PDS periodically checks this table for changes, which need to be synced. Changes are synchronized according to the change-id order.

Each *change-log* table must be used by only a single PDS command. Using a *change-log* table in multiple PDS commands will lead to synchronization errors.

The *change-log* table must have the following structure:

Column name	Data Type	Description
change_id	• 64-bit integer • Primary Key	• Orders the changes. • Non-negative.
record_key	VARCHAR	Record key
parent_key	VARCHAR	Key of parent record, the parent key can be omitted for non-child tables.
Deletion	BIT	Indicates of the record change is an add/update (0), or a deletion (1). 1 – deletion, 0 – add/update
column name	VARCHAR	The name of a column in the RDBMS table. Repeatable, one per RDBMS table column.

If the change-log table has additional columns, PDS ignores them. A timestamp column is a common addition.

State Table

The *state* table can have any table name valid in the source database. This name is entered into the PDS configuration file.

Your application does not need to populate the *state* table. PDS will update it.

The *state* table must have the following structure:

Column name	Data Type	Description
table_name	VARCHAR Primary Key	Name of the RDBMS table
change_id	64-bit integer	The highest change-id that has been synchronized.

Your application can obtain PDS' synchronization state can be obtained by reading the *state* table.

If the *state* table has additional columns, PDS ignores them. However, such columns must have a default value, or PDS can encounter an error updating the *state* table. A timestamp column is a common addition.

A single *state* table can be used by multiple PDS commands.

Erasing a row in the state table or setting the change_id value to 0 causes PDS to re-synchronize all changes for the table named in that row.

Command-Line Usage

The PDS command line is as follows:

```
java [jvm_options] -jar  TIB_tps_datasyncer.jar config-file-name
```

Example:

```
java –jar TIB_tps_datasyncer.jar  names_table_PDS_settings.yaml
```

An example of a shell-script for three joined tables: *persons* (parent table) and *address* and *phone* (child tables).

```
nohup java –jar TIB_tps_datasyncer.jar persons.yaml &> persons_pds.log &
nohup java –jar TIB_tps_datasyncer.jar address.yaml &> address_pds.log &
nohup java –jar TIB_tps_datasyncer.jar phone.yaml   &> phone_pds.log &
```

Logging

When PDS performs logging using Apache™ log4j2, the following events occur:

- The PDS log-level defaults to INFO.
- All other log4j2 defaults are unchanged.

For more information about how to configure logging, see the Apache log4j2 documentation.

Limitations

1. PDS only synchronizes in one direction: from the source database to Patterns.
2. PDS has limited throughput, typically a few thousand records per second. Loading an excessive volume of records can take considerable time.
3. PDS has the tradeoff of latency versus resource-consumption typical of polling applications.
4. PDS commands operate independently of each other. A single PDS command performs synchronizations in the order specified by the *change_id* column in its change-log table. However, when multiple tables are being synchronized using multiple PDS commands, order of synchronization across tables is not guaranteed. Change-log-B can be synced before Change-log-A, even if Change-log-A has earlier changes.

Configuration File

The PDS configuration file contains the following settings:

- Connection settings for the source database
- Connection settings for the Patterns Server
- A map of source-database table columns to Patterns table columns.
- Tuning options

A fully-commented sample configuration file has the following contents:

```
# JDBC Driver
# Only used when the JDBC driver is not pre-packaged.
```

```
# Consult the database vendor's documetation for the class
name.

# jdbcDriver: "RDBMS-specific.
com.company.database.jdbc.SQLServerDriver"

# Connection settings (JDBC parameters) for connecting to the
source database.

# See your database vendor’s documentation
  rdbmsUrl: "RDBMS-specific"
  rdbmsUser: username
  rdbmsPassword: 12345678
  rdbmsSchema: "RDBMS-specific"
  rdbmsCatalog: "RDBMS-specific"

# rdbmsChangeLog
# Name of the source database table that contains the record
changes.
# Case sensitive. Required.
  rdbmsChangeLog: TPS_Addr_change_log

# rdbmsSyncState
# Name of the source database table that contains the sync
state.
# Case sensitive. Required.
  rdbmsSyncState: TPS_sync_state

# Connection settings for the Patterns server

patternsHost
# The IP address or DNS name of the machine where the TIBCO
Patterns is
# running.  Optional. Default “localhost”

patternsHost: localhost
# patternsPort
# The TCP port the TIBCO Patterns Server is listening on.
Optional. Default 5051.
patternsPort: 5051

# patternsSSL
# Whether to use SSL encryption when communicating to the TIBCO
```

```
Patterns Server.# Valid values are any YAML Boolean (e.g. YES
or NO). Optional. Default NO.
patternsSSL: NO

# patternsTableName
# Name of the table in TIBCO Patterns that records will be
synced to.
# Case sensitive. Required.
  patternsTableName: Address

# Map of source-database table columns to Patterns table
columns.

# columns
# Maps source-database table columns to Patterns table columns
# Defines how the RDBMS table columns maps to the Patterns
table columns.
# Case sensitive. Required.
# There must be one entry for each Patterns table column.
# The values are the column-names of the RDBMS table.
# These must be ordered as they appear in the Patterns table.
# Columns in the Patterns table which have no corresponding
# column in the source database are represented by an empty
# string “”.
# For example, if the Patterns table has four columns Fname,
Mname, Lname, Full_name and
# the RDBMS has three columns last, first, full:
columns:
- first        # first -> Fname
- ""           # (unmapped) Mname in Patterns table will always
be blank
- "last"       # last  -> Lname
- "full"       # full  -> Full_name

# Tuning options

# autoPurge
# Policy for purging changes from the RDBMS after they are
synced.
# Purging is performed after an empty batch is encountered.
# Valid values are:
```

```
#     DISABLED: Auto-purge is disabled.
#     ENABLED:  Enables auto-purge only if the Patterns Table
has durable-data enabled.
# This mode may cause performance issues when there is a high
volume of updates.
# FORCE: Enables auto-purge  unconditionally. Not recommended,
it causes
# synchronization issues if Patterns is restarted without an
up-to-date checkpoint.
# Optional. Default DISABLED
  autoPurge: ENABLED

# batchSize
# Number of changes to sync per batch. Optional. Default 100.
batchSize: 100

# interBatchDelay
# Seconds to delay after a non-empty batch of changes.
Optional. Default 0.
# May be fractional. Minimum 0, maximum 300.
interBatchDelay: 0

# emptyBatchDelay
# Seconds to delay after an empty batch of changes. Optional.
Default 30.
# May be fractional. Minimum 1, maximum 300.
emptyBatchDelay: 30.0

# retryDelay
# Seconds to delay before reattempting a failed operation.
Optional. Default 30.
# May be fractional. Minimum 1, maximum 300.
retryDelay: 30.0
```

Note:
You can find a sample (sample_pds_settings.yaml) PDS Yaml file at `tps/6.1/datasync/sample`.

JDBC Support

Data Synchronizer bundles two common JDBC drivers:

- PostgreSQL JDBC Connector, version 42.5.1
- Microsoft JDBC Driver For SQL Server, version 11.2.3.jre8

To use a JDBC driver that is not in the above list:

1. place the driver jar, and any jars it depends on, at the start of the JVM classpath
2. populate the jdbcDriver parameter in the yaml configuration file with the driver class-name.

```
Example
java -classpath ojdbc8.jar;TIB_tps_datasyncer.jar
com.tibco.patterns.datasync.Main yaml-file
where, ojdbc8.jar is the required driver jar file, you must mention the jar
file in classpath.
yaml-file is the PDS configuration YAML file.
```

Note: The JDBC driver must be compatible with the JDK.

Advanced Scenarios

Many-To-One Synchronization

A single Patterns table can be updated from multiple sources by using multiple PDS commands, one per source.

Caution: Ensure that the keys from different sources do not overlap and each key must come from one and only one source. Overlapping keys can lead to loss of updates.

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. 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 1.7 or later.

Command Line Syntax

All commands and arguments are 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

To issue commands from the NSC,:

You must use the NSC.jar located at the following path:

```
TIBCO_HOME/tps/<version>/NSC/bin/
```

Then, run the following OS command line:

```
java -jar NSC.jar [-host hostname] [-port port#][-ssl][-debug] -cmd <-"argument
(s)">
```

NSC Configuration Arguments

Configuration arguments are optional and defaults are used if arguments are 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 has no effect since the secondary commands are already inside the initial connection.

Parameter	Details
`-host <hostname or IP address>`	Specifies the host location where the TIBCO Patterns server is installed. (Default: localhost). You can specify the name or IP address of another machine if the host resides on a remote machine. You cannot use NSC to communicate with a TIBCO Patterns server that does not have IPv4 enabled for input.
`-port <port#>`	Specifies the port to establish a connection on when communicating with the TIBCO Patterns server. (Default: 5051). You can specify a different port if the default is not being used.
`-ssl`	Enables encrypted communications. Use ssl only if the server is configured for it.
`-debug`	Turns on a debug trace of communications between NSC and the TIBCO Patterns server. The debug trace is lengthy and the output is generated to the stderr. 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 can accept a series of commands as follows:

- 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 of a 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
```

Argument	Details
`-repetitions <n>`	Causes the entire batch file to be rerun 'n' number of times. The default value (1) causes the batch file to be issued only once.
`-sleep <msec>`	Results in a pause between –cmd executions. –sleep is specified in milliseconds (1000 milliseconds = 1 second). The default value is 0, which does not delay processing.
-gc	Forces Java garbage collection immediately after the **-cmd** is complete. The default is to let normal garbage collection occur.
`-log <log-file>`	The '-log' argument directs output to the specified file during the batch file execution. The default is for the output to go to stdout and stderr.
`-qa`	Suppresses batch processing informational content such as timestamps.

Table Commands

The following commands are used to interact with the data tables on the matching engine.

tbllist

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

Example output:

```
List: GENERIC
    DBDESCRIPTOR: std
    LAYOUT:
    CHECKPOINT_STATUS: Never Checkpointed
    TRAN_ID: 0
    TRAN_OBJ_STATE: 1
    DBINFO:
    DBNUMFIELDS: 3
    FIELDNAMES: ["FirstName","LastName","Age"]
    FIELDTYPES: [ 5, 5, 5]
    FIELDBYTECOUNTS: [ 66, 74, 26]
    FIELDCHARCOUNTS: [ 66, 74, 26]
    CHARMAPS: ["=STD=","=STD=","=STD="]
    DBGIPFILTER: true
    DBGIPGPU: false
    DBSORTFILTER: false
    DBPSIFILTER: false
    DBNUMRECORDS: 13
    DBKEYTREEKBYTES: 12
    DBRECFIELDKBYTES: 0
    DBHEADERKBYTES: 1323
    DBIDXTOTALKBYTES: 334
    DBTOTALKBYTES: 3741
    PROCTOTALKBYTES: 415268
    TABLE_TYPE: Standard
```

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

The `tblcreate` command is used to create a table. With this command, you can define the following information:

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

- Field indexes.

The `tblcreate` command has two arguments:

- *name* the name of the table
- <*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\st
ate2.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 field name. A field type and a field map must also be specified.

Field Names

Fieldnames must not contain spaces or special characters[1].

Field Types

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

- FLDTYPE_ATTRS - 15
- FLDTYP_SRCHDATE - 11
- FLDTYP_DATE - 10
- FLDTYP_DATETIME - 12
- FLDTYP_FLOAT - 8
- FLDTYP_INT - 6
- FLDTYP_SRCHTEXT - 5

[1]Note this is a restriction of NSC. The server supports blanks and some special characters. For more information about valid field names, see the TIBCO® Patterns Concepts guide.

- FLDTYP_TEXT - 4

Only field types 4 and 5 are searchable using fuzzy search by the matching engine.

character maps

In the event that there is no custom character map to be assigned, the default map '=STD=', should be specified. If a custom map is to be specified, it must be loaded before `tblcreate` is available for table creation. For more information about creating a map, see `-cmd mapcreate`.

Indexes

Indexes are used to speed predicate evaluation. For more information about the creation of indexes, see your TIBCO representative.

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	Description
name	Name of the table to be created. This is required
csv-file	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	Optional arbitrary text associated with this table.

For the **tblload** command, specify **–fieldnamesfirst** in the ***csv-options*** arguments.

csv-options

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

Options	Description
-encoding <*char-set*>	The character set encoding of the file. Valid values are **"latin1**" and "**UTF-8**". The default is "**latin1**".

Options	Description
-domaxwork	If Records in the CSV file with formatting errors or duplicate keys are quietly ignored.
-leadingkey	The first field in each data record is assumed to be the key field for the record. This field is unnamed and does not appear in any header record of field names.
-initialkey <*n*>	If keys are generated automatically the key sequence starts with ***n***. **Default**: Zero.
-fieldnamesfirst	The first line of the CSV file is a header line defining the names of each field. **Default**: No field name header line.
-fieldtypesfirst	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**: No field types header line.
-isparent	The table is created as a parent table. **Default**: Table is not a parent table.
-parenttable <*table-name*>	This is a child table. The *table-name* given must be the name of an existing parent table. It is an error to provide this option if **-isparent** is given. **Default**: Table is not a child table.
-keyfieldindex <*n*>	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.
-keyfieldname <*name*>	The name of the field that is to be used as the record key. This field is not loaded as a data field.
-parentkeyfield <*name*>	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 **–parenttable** 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:

Options	Description
name	Name of the table to be created. This is required.
csv-file	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.

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 of 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>[-local]

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

file is the name of a standard thesaurus definition file. See the TIBCO® Patterns server documentation for the format of this file. If *-local* is supplied, the file is read by NSC from the local file system. Otherwise it is read by the TIBCO Patterns server from its loadable-data directory.

wdcreate -thesaurus <name> -file <file>[-local]

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

file is the name of a weighted dictionary definition file. See the TIBCO® Patterns server documentation for the format of this file. If *-local* is supplied, the file is read by NSC from the local file system. Otherwise it is read by the TIBCO Patterns server from its loadable-data directory.

cthcreate -thesaurus <name> -file <file>[-local]

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. See the TIBCO® Patterns server documentation for the format of this file. If *-local* is supplied, the file is read by NSC from the local file system. Otherwise it is read by the TIBCO Patterns server from its loadable-data directory.

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 recreplace 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:

-mapname <*name*>	the unique identifying name for this character map.
-mapchars <*map-file*>	used to set explicit character mappings. *Map-file* is the name of a file that consists of sets of 3 characters: <***from***><***to***><***new-line***> specifying that character ***from*** is mapped to character ***to*** before comparing strings. The ***new-line*** 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 ***from*** or ***to*** may be the new-line character.)
-foldcase	if this is present all letters are mapped to a common letter case.
-folddiacritics	if present diacritic marks are stripped from letters.
-punctuation	if present all punctuation characters are mapped to the character ***c***.
-whitespace	if present all whitespace characters are mapped to the character **c**.

Index-List 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.

query-str	is the string to be matched.

name	is the name of the table to be searched.
Field1,field2,...	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 an TIBCO® Patterns 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 **debugginon** command in batch commands to turn debug tracing of all interactions with the TIBCO® Patterns server.

debuggingoff

Use the **debugginoff** 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® Patterns 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 theTIBCO® 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 fails.

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 fails.

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

Documentation for TIBCO® Patterns is available on the TIBCO® Patterns Product Documentation page.

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

TIBCO, the TIBCO logo, the TIBCO O logo, ActiveMatrix BusinessWorks, BusinessConnect, TIBCO Hawk, TIBCO Rendezvous, TIBCO Administrator, TIBCO BusinessEvents, TIBCO Designer, and TIBCO Runtime Agent are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT (S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.